Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Araujo Villarinho

A Simheuristic Algorithm for the Stochastic Permutation Flow-Shop Scheduling Problem with Delivery Dates and Cumulative Payoffs

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção.

Advisor : Prof. Luciana de Souza Pessôa
Co-advisor: Prof. Fernando Luiz Cyrino Oliveira

Rio de Janeiro
July 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Pedro Araujo Villarinho

A Simheuristic Algorithm for the Stochastic Permutation Flow-Shop Scheduling Problem with Delivery Dates and Cumulative Payoffs

Dissertation presented to the Programa de Pós–graduação em Engenharia de Produção of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia de Produção. Approved by the Examination Committee.

Prof. Luciana de Souza Pessôa
Advisor
Departamento de Engenharia Industrial – PUC-Rio

Prof. Fernando Luiz Cyrino Oliveira
Co-advisor
Departamento de Engenharia Industrial – PUC-Rio

Prof. Helena Ramalhinho Dias Lourenço
Universitat Pompeu Fabra – UPF

Prof. Rafael Martinelli Pinto
Departamento de Engenharia Industrial – PUC-Rio

Rio de Janeiro, July the 16th, 2020

Pedro Araujo Villarinho

Majored in Industrial Engineering by Pontifical Catholic University of Rio de Janeiro - PUC-Rio

Bibliographic data

Villarinho, Pedro Araujo

A Simheuristic Algorithm for the Stochastic Permutation Flow-Shop Scheduling Problem with Delivery Dates and Cumulative Payoffs / Pedro Araujo Villarinho; advisor: Luciana de Souza Pessôa; co-advisor: Fernando Luiz Cyrino Oliveira. – Rio de janeiro: PUC-Rio, Departamento de Engenharia Industrial. - 2020.

v., 57 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2020.

Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Problema de Programação de Máquinas em Série;. 3. Heurísticas;. 4. Simulação;. 5. Métricas de risco.. I. Pessôa, Luciana de Souza. II. Cyrino Oliveira, Fernando Luiz. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

To my family.

Acknowledgments

To my family, in special, to my mother Fernanda, my father Alvaro, my brother Gabriel for the unconditional support.

I would like to thank my advisor Luciana Pessôa and my co-advisor Fernando Cyrino, for all the support, learning, ideas, patience, opportunities, motivation, guidance, and friendship during my master's.

I am also thankful to Professor Bruno Fanzeres for all the enriched discussion and help whenever I need.

I wish to thank Professor Angel Juan and Professor Javier Panadero, for receiving me as visiting student at Internet Computing Systems Optimization (ICSO) - UOC Barcelona, for the opportunity, for all the enriching discussions, guidance, patience, ideas, and friendship that have certainly assisted on the development of this and other works.

To all the Professors, colleagues, and employees of the Industrial Engineering Department (DEI) for all the knowledge and support.

To all of my friends.

To PUC-Rio and the Industrial Engineering Department (DEI) for the opportunity.

This work was partially supported by own financial resources, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

This study was financed in part by the Coordenação de Aperfeiçoaento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Villarinho, Pedro Araujo; Pessôa, Luciana de Souza (Advisor); Cyrino Oliveira, Fernando Luiz (Co-Advisor). **A Simheuristic Algorithm for the Stochastic Permutation Flow-Shop Scheduling Problem with Delivery Dates and Cumulative Payoffs**. Rio de Janeiro, 2020. 57p. Dissertação de mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

This master's thesis analyzes the Permutation Flow-shop Scheduling Problem with Delivery Dates and Cumulative Payoffs under uncertainty conditions. In particular, the work considers the realistic situation in which processing times and release dates are stochastics. The main goal is to solve this Permutation Flow-shop problem in the stochastic environment and analyze the relationship between different levels of uncertainty and the expected payoff. In order to achieve this goal, first a biased-randomized heuristic is proposed for the deterministic version of the problem. Then, this heuristic is extended into a metaheuristic by encapsulating it into a variable neighborhood descent framework. Finally, the metaheuristic is extended into a simheuristic by incorporating Monte Carlo simulation. According to the computational experiments, the level of uncertainty has a direct impact on the solutions provided by the simheuristic. Moreover, a risk analysis is performed using two well-known metrics: the value at risk and the conditional value at risk.

Keywords

Permutation Flow-shop Scheduling Problem; Heuristic; Simulation; Risk-metrics.

Resumo

Villarinho, Pedro Araujo; Pessôa, Luciana de Souza; Cyrino Oliveira, Fernando Luiz. **Um algoritmo de sim-heuristica para um problema estocástico de Permutation Flow-shop Scheduling com datas de entrega e ganhos cumulativos**. Rio de Janeiro, 2020. 57p. Dissertação de Mestrado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação de mestrado analisa um problema de programação de máquinas em série com datas de entrega e ganhos cumulativos sob incerteza. Em particular, este trabalho considera situações reais na quais os tempos de processamento e datas de liberação são estocásticos. O objetivo principal deste trabalho é a resolução deste problema de programação de máquinas em série em um ambiente estocástico buscando analisar a relação entre diferentes niveis de incerteza e o benefício esperado. Visando atingir este objetivo, primeiramente uma heurística é proposta utilizando-se da técnica de *biased-randomization* para a versão determinística do problema. Então, esta heurística é extendida para uma metaheurística a partir do encapsulamento dentro da estrutura de um *variable neighborhood descend*. Finalmente, a metaheurística é extendida para uma simheurística a partir da incorporação da simulação de Monte Carlo. De acordo com os experimentos computacionais, o nível de incerteza tem um impacto direto nas soluções geradas pela simheurística. Além disso, análise de risco foram desenvolvidas utilizando as conhecidas métricas de risco: *value at risk* e *conditional value at risk*.

Palavras-chave

Problema de Programação de Máquinas em Série; Heurísticas; Simulação; Métricas de risco.

Table of contents

1 Introduction **13**

2 Problem Definition **15**
2.1 Deterministic PFSPDP 15
2.1.1 Formal Definition 17
2.2 Stochastic PFSPDP 18

3 Related Work **21**
3.1 The Deterministic PFSP with Different Time-Related Goals 21
3.2 The Stochastic PFSP 22

4 Proposed Methods **25**
4.1 A Biased-Randomized Algorithm for the PFSPDP 25
4.1.1 The FF Heuristic 25
4.1.2 Extending the FF Heuristic to a Biased-Randomized Algorithm 26
4.1.3 Extending the BR-FF Algorithm to a Metaheuristic 27
4.2 Extending the BR-FF-VND Algorithm to a Simheuristic 28
4.3 Risk Metrics 33

5 Results and Discussion **34**
5.1 Experimental Settings 34
5.1.1 Benchmark Instances 34
5.1.2 Computational Environment & Parameter Settings 35
5.2 Testing the Biased-Randomized Algorithm in the Deterministic Scenario 36
5.3 Testing a Simheuristic in the Stochastic Scenario 39

6 Conclusions **46**

Bibliography **48**

List of figures

Figure 2.1 Example of a flow-shop scheduling with delivery dates. 16
Figure 2.2 Example of a flow-shop scheduling with delivery dates with stochastic times. 19
Figure 2.3 Example of a flow-shop scheduling with delivery dates with stochastic times and stochastic release date. 20

Figure 5.1 Deviation of best solutions between the BR-FF and FF heuristics versus the β parameter. 36
Figure 5.2 Box-plot of the Percentage deviation *AvgDev (%)* and BestDev (%). 38
Figure 5.3 Percentage deviation of *BestDev*(%) and *AvgDev*(%) to BR-FF-VND in the deterministic environment. 39
Figure 5.4 Percentage deviation of the best solutions - *BestDev*(%) between the Sim BR-FF-VND and the deterministic BR-FF-VND 1.000 times simulated. 41
Figure 5.5 Conditional VaR for instance *k10a0.3r0.1* using different levels of α and $h = 1.0$. 42
Figure 5.6 Behavior of the $\text{VaR}_{95\%}$ and $\text{CVaR}_{95\%}$ on different levels of uncertainty for instance *k10a0.1r0.3*. 42
Figure 5.7 Percentage deviation of *BestDev* (%) and *AvgDev* (%) to BR-FF-VND in the deterministic environment. 43
Figure 5.8 Behavior of the $\text{VaR}_{95\%}$ and $\text{CVaR}_{95\%}$ on different levels of uncertainty for instance *k10a0.1r0.3* in a PSFPSTRD 44
Figure 5.9 Percentage deviation of *BestDev*(%) and *AvgDev*(%) to SimBR-FF-VND in PFSPSTRD 45

List of tables

Table 5.1 A comparison of algorithms for 1 minute computational time. 37
Table 5.2 A comparison of algorithms for different computing times. 38

List of Abreviations

PFSPDP – Permutation Flow-shop Scheduling Problem with with Delivery Dates and Cumulative Payoff
PFSPDPST - Permutation Flow-shop Scheduling Problem with Delivery Dates, Cumulative Payoff and Stochastic Times
PFSPSTRD - Permutation Flow-shop Scheduling Problem with Delivery Dates, Cumulative Payoff, Stochastic Times and Release Dates.
BR – Biased Randomization
MCS – Monte Carlo simulation
PFSP – Permutation flow-shop problem
FSP – Flow-shop Scheduling Problem
GA – Genetic Algorithm
GRASP – Greedy randomized adaptive search procedure
BRKGA – Biased random-key genetic algorithm
VND - Variable Neighborhood descent
ILS – Iterated Local Search
VaRα – Value at Risk for a level of confidence α
CVaRα –Conditional Value at Risk for a level of confidence α

Seja você quem você for, seja qualquer posição que você tenha na vida, do nível altíssimo ao mais baixo social, tenha sempre como meta muita força, muita determinação e sempre faça tudo com muito amor e com muita fé em Deus, que um dia você chega lá. De alguma maneira você chega lá.

Ayrton Senna, *1960 - 1994.*

1
Introduction

In the Permutation Flow-shop Scheduling problem with Delivery Dates and Cumulative Payoff (PFSPDP), a finite number of jobs j is processed – following the same order from a release date r_j – by a finite number of machines m. Each time a job finishes before a delivery date, a reward is obtained. Hence, the main goal is to find the permutation of jobs that maximizes the total reward related to the jobs completion times. Typically, job rewards are assigned based on a decreasing step-wise function, i.e., delivery dates define a series of time intervals, and the sooner a job is completed the greater the associated reward. This problem was first studied by Seddik et al. [1], who modeled a book digitization service in a one-machine environment. Then, Pessoa and Andrade [2] extended the aforementioned problem into a flow-shop context, and proposed a mathematical formulation and a set of heuristic and metaheuristic methods.

This research takes one step forward and proposes a biased-randomized (BR) algorithm (Grasas et al. [3]) based on the FF heuristic developed by Fernandez-Viagas and Framinan [4]. The BR-FF heuristic has shown a remarkable performance when solving the benchmarks proposed in Pessoa and Andrade [2]. This is coherent with previous results in which biased-randomized algorithms have shown to be competitive in different flow-shop problems (Ferrer et al. [5]; Ferone et al. [6]).

Since uncertainty is often present in many services and production systems, the main motivation of this research is to solve this PFSPDP in the stochastic environment, aiming at answering two research question: (*i*) the behavior of the reward in different scenarios of uncertainty and (*ii*) the worst expected scenarios concerning the reward in each different scenario of uncertainty, to do it so two stochastic versions were proposed. First, the PFSPDP was studied in an environment in which the processing times were considered as random variables, named PFSPDP with Stochastic Times (PFSPDPST). Then, the aforesaid version was extended to a new one in which not only considers the uncertainty in the processing times of each job in each machine but also considers the uncertainty present in the release date of each job in the first machine, named PFSPSP with Stochastic Times and Release

Dates (PFSPSTRD).

In order to solve those challenging versions, the previous BR-FF algorithm is first integrated into a variable neighborhood descent (VND) metaheuristic framework. Then, this BR-FF-VND metaheuristic is extended into a simheuristic algorithm (Juan et al. [7]) by combining the VND metaheuristic with Monte Carlo simulation (MCS). In fact, simheuristics have been successfully used in the recent past to cope with stochastic versions of different flow-shop problems (Gonzalez-Neira et al. [8]; Hatami et al. [9]). Another significant contribution of this work relates to the use of two risk metrics: the value at risk (VaRα) and the conditional value at risk (CVaRα). They are used to complement the simheuristics approach, offering the decision-maker an analysis of the worst expected cases concerning the rewards.

This work is divided into several Chapters. Chapter 2 contains a detailed description of the problem under consideration. A literature review on related work is provided in Chapter 3. In Chapter 4 the proposed methods are described, to do it so, Section 4.1 describes a biased-randomized algorithm for solving the deterministic version of the problem, as well as its extension into a metaheuristic. Section 4.2 extends the previous metaheuristic into a simheuristic to cope with the stochastic versions of the problem. In Chapter 5 the settings of the computational experiments are provided in Section 5.1, while the actual experiments are provided in Sections 5.2 (deterministic variant) and 5.3 (stochastic and risk variant). Finally, Chapter 6 highlights the main contributions of this work and discusses some open research lines.

2
Problem Definition

In this Chapter the problem carried out in this work will be detailed in two environments deterministic and stochastic.

Aiming at contextualizing this PFSPDP in a real-life context to a better comprehension of the reader, the environment of book digitization studied by Seddik et al. [1] and extended by Pessoa e Andrade [2] will be address in this research. However, it is important to highlight that this PFSP could be addressed in other environments, which aims at finishing the process as soon as possible, for instance in the retailing when the company wants to release the product as soon as possible for their customer.

As described in the previous references, the problem is motivated by a book-digitization project in a library. The book digitization is a job and this has to be processed in two consecutive stages (machines): digitization and segmentation. Each book requires a given amount of time to be processed in each stage. Due to the operational constraints in the library, not all books are available for digitization at the same time. Hence, a release date is set for each one. This problem is modeled by taking into account two entities: the library – which establishes the delivery dates –, and the digitization firm. Each delivery date is a reference point for computing the payment: the sooner each book is processed the higher the corresponding reward. This research consider the problem from the point of view of the digitization firm, which wants to maximize its profits.

Therefore, the hired company wants to identify the best permutation of books that provides the hired rewards, and the worst expected scenarios of rewards.

2.1
Deterministic PFSPDP

The deterministic version of the PFSPDP is described in this Section. A finite set of n jobs, $\mathcal{J} = \{1, 2, \ldots, n\}$ is processed in a subsequent way by m machines $\mathcal{M} = \{1, 2, \ldots, m\}$. For each job $j \in \mathcal{J}$ and machine $l \in \mathcal{M}$, the processing time of job j in machine l is a known constant $p_{jl} > 0$ (notice that no uncertainty is considered in this deterministic version

of the problem). Furthermore, let S (schedule) be a finite set containing the times s_{jl} at which each job j starts being processed at each machine l, i.e., $S = \{s_{11}, \ldots, s_{1m}, s_{21}, \ldots, s_{2m}, \ldots, s_{n1}, \ldots, s_{nm}\}$. Given a schedule S, the completion time of a job j in a machine l is given by $c_{jl} = s_{jl} + p_{jl}$. For simplicity, let c_j denote the completion time of job j in the last machine $m \in \mathcal{M}$, i.e., $c_j = c_{jm}$.

The problem addressed in this work also considers release dates, $r_j \geq 0$, in which job j becomes available to start being processed by machine 1. Additionally, a set of k delivery dates $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ is given. Each job j has an associated reward, $\mathcal{F}(c_j)$, which depends upon the time it is completed:

$$\mathcal{F}(c_j) = \begin{cases} 0 & \text{if } \; d_k < c_j \\ 1 & \text{if } \; d_{k-1} < c_j \leq d_k \\ \vdots & \\ k & \text{if } \; 0 < c_j \leq d_1 \end{cases} \tag{2-1}$$

Hence, the goal is to obtain a permutation of jobs that maximizes the aggregated reward obtained, i.e., the objective function is: $\max \sum_{j=1}^{n} \mathcal{F}(c_j)$.

To illustrate how the reward function works, consider the following simple example (Figure 2.1):

- Number of machines $m = 2$, number of jobs $n = 4$.
- Release dates: $r_1 = 14$, $r_2 = 2$, $r_3 = 9$, and $r_4 = 7$.
- Processing times: $p_{1l} = (5, 1)$, $p_{2l} = (3, 2)$, $p_{3l} = (3, 6)$, and $p_{4l} = (6, 3)$.
- Delivery dates: $d_1 = 10$, $d_2 = 20$.

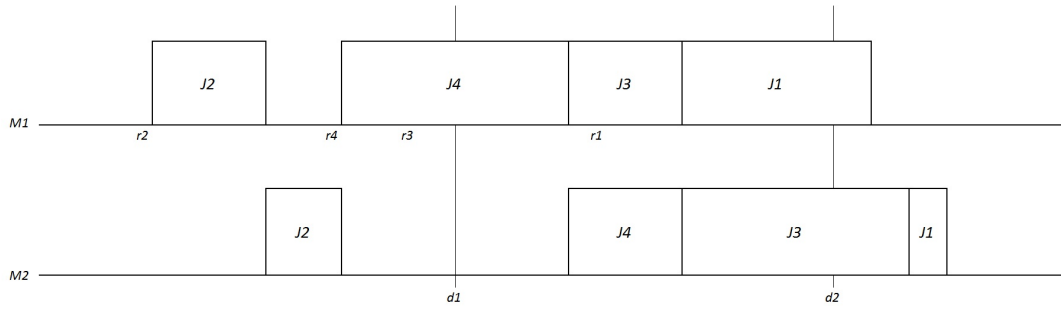

Figure 2.1: Example of a flow-shop scheduling with delivery dates.

For this instance, which contains 2 delivery dates, the reward step-wise function has the following configuration:

$$\mathcal{F}(C_j) = \begin{cases} 0 & \text{if } d_2 < c_j \\ 1 & \text{if } d_1 < c_j \leq d_2 \\ 2 & \text{if } 0 < c_j \leq d_1 \end{cases} \tag{2-2}$$

As illustrated in Figure 2.1, since job 2 has a completion time $c_2 = 7$, which is lower than delivery date $d_1 = 10$, there is an associated reward of 2 units. In addition, job 4 has a completion time $c_4 = 16$, which is lower than delivery date $d_2 = 20$, thus generating a reward of 1 units. Also, since jobs 3 and 1 have completion times beyond the delivery date $d_2 = 20$ ($c_3 = 22$ and $c_1 = 23$), no associated rewards are obtained. All in all, the accumulated reward in this case is 3 units.

2.1.1 Formal Definition

The formal definition of the problem follows the proposed definition in Pessoa and Andrade [2] for a Mixed-Integer Linear Program (MILP) to model this PFSP.

Hence, consider the binary decision variable X_{ij} as $X_{ij} = 1$ if job J_i is the j^{th} job of the sequence and $X_{ij} = 0$ if not. Variable $C^{pos}_{j\ell} \in \mathbb{Z}^+$ denotes the completion time of the job on the j^{th} position on machine M_ℓ for $J_j \in \mathcal{J}$ and $M_\ell \in \mathcal{M}$. The $F|r_j, perm| \sum_{j=1}^{n} \mathcal{F}(C_j)$ can be formulated as following:

$$\max \quad \sum_{j=1}^{n} \mathcal{F}(C^{pos}_{jm}) \tag{2-3}$$

$$\text{s.t.} \sum_{i=1}^{n} X_{ij} = 1 \; \forall j \in \{1, \ldots, n\}, \tag{2-4}$$

$$\sum_{j=1}^{n} X_{ij} = 1 \; \forall i \in \{1, \ldots, n\}, \tag{2-5}$$

$$C^{pos}_{j1} \geq \sum_{i=1}^{n} (p_{i1} + r_i) X_{ij} \; \forall j \in \{1, \ldots, n\}, \tag{2-6}$$

$$C^{pos}_{j\ell} \geq C^{pos}_{j-1,\ell} + \sum_{i=1}^{n} p_{i\ell} X_{ij} \; \forall j \in \{2, \ldots, n\} \; \forall \ell \in \{i, \ldots, m\}, \tag{2-7}$$

$$C^{pos}_{j,\ell+1} \geq C^{pos}_{j\ell} + \sum_{i=1}^{n} p_{i,\ell+1} X_{ij} \; \forall j \in \{2, \ldots, n\} \; \forall \ell \in \{1, \ldots, m-1\}, \tag{2-8}$$

$$C^{pos}_{j\ell} \geq 0 \; \forall j \in \{2, \ldots, n\} \; \forall \ell \in \{i, \ldots, m\}, \tag{2-9}$$

$$X_{ij} \in \{0, 1\} \; \forall i, j \in \{1, \ldots, n\}. \tag{2-10}$$

The Objective Function (2-3) seeks to maximize sum of the rewards. Constraints (2-4) and (2-5) ensure that each job is allocated in one position of the scheduling and vice-versa. Constraint (2-6) establishes the job J_j in the first

can only start when their release date is met or after it. Constraint (2-7) avoids the overlapping of a job and its predecessor on a machine. Constraint (2-8) guarantees that a job can only be processed on a machine after it has finished on the former one. Constraints (2-9) and (2-10) give the definition domain of the variables.

2.2 Stochastic PFSPDP

In the stochastic version of the PFSPDP, a finite set of n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$, are processed by m machines $\mathcal{M} = \{1, 2, \ldots, m\}$ following in the same order. For each job $j \in \mathcal{J}$ and machine $l \in \mathcal{M}$, the processing time of job j in each machine l is not-known beforehand and follows a non-negative probability distribution $p_{jl} > 0$. The stochastic version of this problem has the same constraints as the deterministic version, which has been defined in Section 2.1, and aims at obtaining a permutation of jobs that maximizes the expected reward obtained.

To do it so, an assumption of a strong correlation between the best deterministic solution and the best stochastic solution, in moderate levels of uncertainty, i.e., lower and medium levels of variance, is taken into account (Juan et al. [10]). Therefore, is it possible to consider that the scheduling provided by the deterministic environment is the same provided by the stochastic environment.

However, it is important to highlight, that the permutation of jobs that generate the best solution on the deterministic environment not necessarily will be the same permutation of jobs that generate the best solution in the stochastic environment since the reward calculated by step-wise function defined on Section 2.1 is directly impacted by the variations on the processing times.

Figure 2.2 portrays the PFSPDPST, in which, the same permutation of jobs illustrated in Section 2.1 is used, following the instances defined bellow, which considers increases and reductions on processing times.

- Number of machines $m = 2$, number of jobs $n = 4$.
- Release dates: $r_1 = 14$, $r_2 = 2$, $r_3 = 9$, and $r_4 = 7$.
- Processing times: $p_{1l} = (4, 2)$, $p_{2l} = (4, 5)$, $p_{3l} = (8, 4)$, and $p_{4l} = (6, 9)$.
- Delivery dates: $d_1 = 10$, $d_2 = 20$.

While on the deterministic environment, i.e, with fixed and known beforehand processing times, the reward generated is equal to 3 units as it

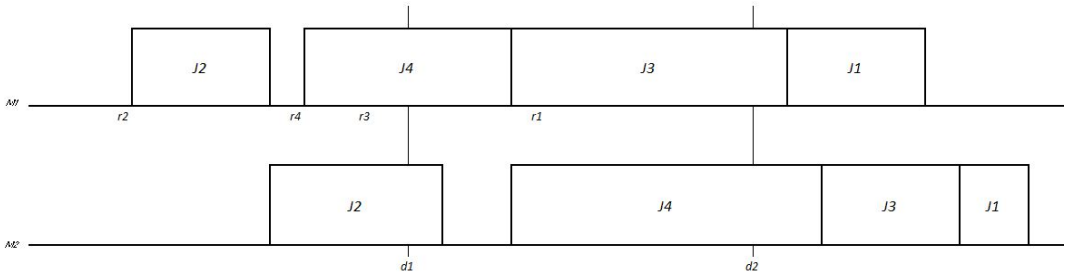

Figure 2.2: Example of a flow-shop scheduling with delivery dates with stochastic times.

was discussed in Section 2.1, on the stochastic environment, increases and reductions on the processing times, directly impact on the reward generated.

On this permutation of jobs portrayed on Figure 2.2, it is notice that there are no jobs finished on the machine M_2 before the delivery date d_1. Moreover, there is just one job J_2 with completion time on the machine M_2 ($C_2 = 11$) between the delivery dates d_1 and d_2, having all the others jobs finished after the delivery date $d_2 = 20$. Therefore, the expected reward generated by this permutation of jobs on the stochastic environment is equal to 1 unit.

Aiming to study this PFSPDP in a more realist environment, another stochastic scenario is approached in this work.

Since in real life the availability of a job j could be affected by uncertainties it is reasonable to consider their variation in the problem's formulation.

To do it so, the assumption of the release date r_j is not known in advance and follows a non-negative probability distribution $r_j > 0$ has been taken into account, making this Permutation Flow-shop Scheduling Problem with Stochastic Times (PFSPDPST) extended to a Permutation Flow-shop Scheduling Problem with Stochastic Times and Release Dates (PFSPSTRD).

Figure 2.3 illustrates the PFSPSTRD, following the same permutation of jobs illustrated in Section 2.1, and whose instance is defined bellow, considering variation in the processing times and release dates.

- Number of machines $m = 2$, number of jobs $n = 4$.
- Release dates: $r_1 = 18$, $r_2 = 1$, $r_3 = 11$, and $r_4 = 7$.
- Processing times: $p_{1l} = (4, 2)$, $p_{2l} = (4, 5)$, $p_{3l} = (8, 4)$, and $p_{4l} = (6, 9)$.
- Delivery dates: $d_1 = 10$, $d_2 = 20$.

As it could be notice in Figure 2.3 the job J_2 has completion time on the machine M_2 ($C_2 = 10$) equal to the delivery date d_1, having received the associated reward of 2 units. Furthermore, there are no jobs with completion

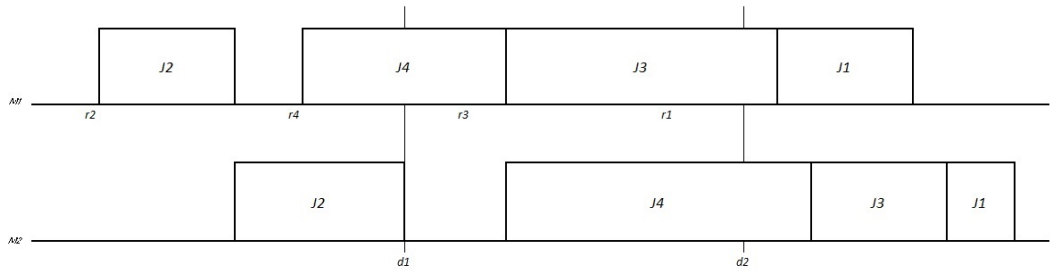

Figure 2.3: Example of a flow-shop scheduling with delivery dates with stochastic times and stochastic release date.

time on the machine M_2 between the delivery dates d_1 and d_2, having all the remaining jobs finished after the delivery date $d_2 = 20$. Hence, the expected reward generated by this permutation of jobs on the stochastic environment is equal to 2 units.

Those aforementioned Permutation Flow-shop Scheduling Problems (PFSP), are examples that there is a need to identify and build solutions to deal with uncertainty environments by finding good solutions in a reasonable time.

3
Related Work

The simplest version of the problem described in Section 2.1 was initially addressed by Seddik et al. [1]. Their work is inspired by a real-life case related to the book digitalization with delivery dates. These authors consider a single machine, being the problem known as the $1|r_j|\sum_{j=1}^{n} F(c_j)$ as defined by Graham et al. [11]. Later, Pessoa and Andrade [2] extended the model to consider a flow-shop environment with m machines, i.e.: $F|r_j, perm|\sum_{j=1}^{n} F(c_j)$. This Section reviews some related work on the PFSP, both in its deterministic and stochastic versions.

3.1
The Deterministic PFSP with Different Time-Related Goals

Since the first formulation proposed by Johnson [12], which aimed at minimizing the makespan in two machines, other formulations were developed on the Flow-shop Scheduling Problem (FSP) as well as heuristic and metaheuristic methods. For complete reviews on deterministic FSP employing the makespan minimization criterion, readers are referred to Framinan et al. [13] and Fernandez-Viagas et al. [14]. For an overview of FSP, readers are referred to Gupta and Stafford Jr [15]. Despite the makespan minimization has been the most popular goal in FSP, other objective functions have been considered as well. Thus, Framinan and Leisten [16] proposed a new greedy algorithm based on the variable neighborhood search (Mladenovic and Hansen [17]) to minimize the total tardiness. Regarding to the minimization of total flow time, Liu and Reeves [18] presented the LR constructive. Similarly, Framinan and Leisten [19] proposed another constructive heuristic for the same purpose. Also, Framinan et al. [20] compared several heuristics and proposed two new ones, which have shown to be efficient to minimize the total flow time. A bi-objective problem considering the minimization of the makespan and the total flow time was tackled by Pasupathy et al. [21]. These authors proposed a genetic algorithm (GA) to construct a Pareto frontier. Nagano and Moccellin [22] aimed at minimizing the flow time by proposing a new heuristic based on the well-known NEH heuristic (Nawaz et al. [23]) and some local search procedures. Also aiming at minimizing the total flow time, an iterated local

search was proposed by Dong et al. [24]. Tasgetiren et al. [25] expanded the discrete differential evolution algorithm proposed by Pan et al. [26] into a hybrid version, and proposed a discrete artificial bee colony algorithm aimed at minimize the total flow time. Della Croce et al. [27] proposed a metaheuristic algorithm for minimizing the total completion time in a two-machine flow-shop problem. Pan and Ruiz [28] made a comprehensive evaluation of 22 heuristics using as objective function the minimization of the total flow time. Fernandez-Viagas and Framinan [4] proposed a new heuristic, named FF, based on the LR heuristic proposed by Liu and Reeves [18]. Lee and Chung [29] addressed an m-machine flow-shop problem with learning effects. The goal was to minimize the total tardiness by proposing a branch-and-bound method and two heuristics. Molina-Sánchez and González-Neira [30] introduced a greedy randomized adaptive search procedure (GRASP) to tackle the minimization of the total weighted tardiness.

Also to minimize the flow time, Abedinnia et al. [31] introduced a constructive heuristic based on the one proposed by Laha and Sarin [32]. Yu and Seif [33] proposed a lower bound based on a GA to minimize the total tardiness and maintenance cost. Rossi et al. [34] sought to minimize the total flow time by employing a new constructive heuristic based on the FF-NEH heuristic introduced by Fernandez-Viagas and Framinan [4]. With the aim of minimizing the total completion time, Fernandez-Viagas and Framinan [35] proposed a beam-search-based constructive heuristic. The minimization of total tardiness was tackled by Fernandez-Viagas et al. [36]. More recently, Rossi and Nagano [37] tackled a mixed no-idle PFSP with sequence-dependent setup times. With the goal of minimizing the total flow time, they proposed a new set of heuristics based on the beam-search algorithm. Likewise, several matheurisitc algorithms were proposed and compared to a GA by Ta et al. [38], with the purpose of minimizing the total tardiness. Finally, Andrade et al. [39] proposed a biased random-key genetic algorithm (BRKGA), with a feature called shaking, to minimize the total completion time.

3.2 The Stochastic PFSP

Considering the great complexity in troubleshooting NP-hard problems, the difficulty of making decisions on time and the fact that those decisions are taken in the real world, where the behavior of its parts is not deterministic, it is clear the need to identify and develop techniques and tools that allow the balance between good quality solutions and appropriate execution times.

In the real world, unforeseen events are often present in daily-life situ-

ations, as in the service and production systems which are subject to uncertainties as, for example: machine failures, imprecise processing times, employee absences, material unavailability, variations in demand, rush orders and order cancellations, changes in due dates, etc. (Elyasi and Salmasi [40]). The literature on flow-shop problems with stochastic components is not as extensive as in the case of the deterministic version. Thus, while in the deterministic version of the PFSP one assumes that the processing time of a job j in a machine l, $p_{jl} > 0$, is known in advance and cannot change over time, in the stochastic version it is usual to consider this processing time as a random variable, P_{jl}, following a non-negative probability distribution (Gonzalez-Neira et al. [41]). By modeling processing times this way, it is possible to represent the uncertainty that is present in real-life scenarios. However, the inclusion of random variables in the optimization problem also increases the difficulty of solving it (Baker and Altheimer [42]).

Banerjee [43] and Makino [44] were among the pioneers in the study of the PFSP with stochastic processing times. The former aimed at minimizing the expected of lateness in a single machine problem, with random processing times following a known probability distribution. In order to do it, the author proposed a decision rule for the single machine problem. The latter author aimed at minimizing the expected makespan considering that processing times follow exponential or k-Erlang probability distributions. He also proposed a sequence rule to find a solution in a scenario with two jobs and three machines. This work was expanded by Talwar [45], who aimed at minimizing the idle time of the last machine by considering the processing times as random variables following exponential probability distributions. Likewise, Cunningham and Dutta [46] considered the processing times being exponentially distributed to minimize the expected makespan in a problem with two machines. Mittal and Bagga [47] also aimed at minimizing the expected makespan in a problem with exponentially distributed processing times. Dodin [48] sought to minimize the expected makespan by representing processing times as random variables following different probability distributions (uniform, Normal, and exponential). Kamburowski [49] drew up an approach aimed at minimizing the makespan in a scenario with two machines and independent processing times. Later, Kamburowski [50] expanded the previous work to a scenario involving three machines. Gourgand et al. [51] completed a review about the stochastic flow-shop scheduling problem. Wang et al. [52] also sought to minimize the makespan while modeling processing times using uniform probability distributions and employing a GA. With the same goal,Kalczynski and Kamburowski [53] modeled processing times using Weibull probability distri-

butions. Baker and Trietsch [54] developed three heuristic procedures for the two-machine permutation flow-shop problem with stochastic processing times. Choi and Wang [55] used a Gamma probability distribution to model processing times. Liefooghe et al. [56] proposed an indicator-based evolutionary algorithm to deal with a bi-objective PFSP with stochastic processing times. Baker and Altheimer [42] evaluated and compared several heuristic procedures for the stochastic version of the problem. Juan et al. [10] and Framinan and Perez-Gonzalez [57] targeted the minimization of the expected makespan by modeling processing times as random variables following log-Normal probability distributions. Gonzalez-Neira et al. [58] presented an overview of PFSP variants with stochastic components. Gholami-Zanjani et al. [59] introduced robust optimization and fuzzy optimization to model the uncertainty of the input data, and minimized weighted mean completion time in a PFSP considering set-up and non-deterministic processing times. Cui et al. [60] analyzed a bi-objective problem including the quality robustness and solution robustness in environments subject to failure uncertainty, which was modeled using Weibull probability distributions. Recently, Framinan et al. [61] explored the use of real-time information to job rescheduling in a PFSP with stochastic processing times.

In this context, this work contributes to the literature by proposing a new heuristic and metaheuristic methods to deal with PFSPDP. Furthermore, this research extends the aforesaid proposed metaheuristic into a simheuristic to solve PFSPDP in two stochastic environments (PFSPDPST and PFSPSTRD) and presents a risk metrics analysis to complement the analyses of the simheuristic results using the well-known risk metrics VaRα and CVaRα.

4
Proposed Methods

This Chapter detailed the proposed methods used to solve the PFSPDP in the deterministic and stochastic environments defined in Chapter 2.

4.1
A Biased-Randomized Algorithm for the PFSPDP

In this Section, a biased-randomized algorithm is proposed for the deterministic version of the PFSP with delivery dates and cumulative payoff. The algorithm is based on the constructive heuristic FF by Fernandez-Viagas and Framinan [4]. This heuristic was also examined by Pessoa and Andrade [2], who showed its superior performance when compared to other constructive procedures for the PFSPDP.

4.1.1
The FF Heuristic

The FF heuristic proposed by Fernandez-Viagas and Framinan [4] aiming at minimizing the total completion time in a permutation flow-shop problem, can be considered as an improved version of the LR heuristic proposed by Liu and Reeves [18]. A solution is built by attaching, at each iteration, an unscheduled job in the last position of a partial solution, according to the index function ξ' given in Equation 4-1.

At each iteration, the index function ξ' is computed considering the weighted idle time, IT'_{jk}, between the last job in the partial solution – being represented as k – and each remaining job outside the partial solution – being represented as j. Moreover, the index ξ' approximate the makespan of the solution (AT'_{jk}).

$$\xi'_{jk} = \frac{(n-k-2)}{a} IT'_{jk} + AT'_{jk} \tag{4-1}$$

where AT'_{jk} and IT'_{jk} are defined as follows:

$$IT'_{jk} = \sum_{i=2}^{m} \frac{m * \max\{c_{(i-1)j} - c_{i[k]}, 0\}}{i - b + k(m-i+b)/(n-2)} \tag{4-2}$$

$$AT'_{jk} = c_{mj} \tag{4-3}$$

The parameters a and b weight the idle time and makespan in the ξ' function, n represents the total number of jobs and k is the total of jobs in the partial solution. After ξ' is computed, the jobs are sorted according to the ascending order of its ξ' value. Then, the job with the smallest ξ' value is scheduled in the partial solution. In case of ties, the job with the smallest value of IT'_{jk} is selected.

Pessoa and Andrade [2] proposed a slight modification of the idle time (IT'_{jk}) for the first machine, to consider the job's release dates (r_j). In the new expression, $\max\{c_{(i-1)j} - c_{i[k]}, 0\}$ was replaced by $\max\{r_j - c_{i[k]}, 0\}$.

Regarding the PFSPDP, the authors have shown the superior performance of the FF heuristic when compared to other constructive heuristics, such as the release dates (R) heuristic (Potts [62]) in which a left-shifted scheduling is provided by sorting the jobs in non-decreasing release date; The earliest completion time (ECT) heuristic (Ladhari and Rakrouki [63]) in which jobs are sorted in increasing order by the earlier possible completion time aiming at providing a left-shifted scheduling; The classical NEH heuristic (Nawaz et al. [23]), and the iterated earliest completion time (IECT) heuristic (Pessoa and Andrade [2]) which follows the principles of the NEH and ECT methods.

4.1.2 Extending the FF Heuristic to a Biased-Randomized Algorithm

In this research, a biased-randomized extension of the FF heuristic, called BR-FF is proposed. As explained in detail by Grasas et al. [3], biased-randomization techniques can be used to incorporate an oriented (non-uniform) random behavior into a base heuristic. This random behavior, typically achieved with the use of a skewed probability distribution, allows to encapsulate the modified constructive heuristic into a multi-start framework, thus exploring alternative paths during the solution-building process. Still, due to the oriented nature of the randomization process, each of these paths follows the logic behind the heuristic – i.e., the most 'promising' movements receive a higher probability of being selected during the constructive process. As a result, each time the biased-randomized heuristic is executed, chances are that the emerging solution outperforms the one provided by the original heuristic (Faulin et al. [64]). There is an additional benefit of applying biased-randomization techniques to a constructive heuristic: the resulting procedure is able to quickly generate different alternative solutions of reasonably good quality (Juan et al. [65]).

In this work, the biased-randomization effect has been applied to the job selection process. Thus, while in the original FF heuristic the job with

the smallest ξ' value is always selected, in the proposed BR-FF a diminishing probability of being selected is assigned to each possible job in the list of jobs sorted from lower to higher ξ' value. This is achieved by employing a geometric probability distribution with a single parameter β ($0 < \beta < 1$). Hence, the random behavior of the selection process depends on the value of β. As this value converges to 1, the selection behavior becomes more greedy (i.e., as in the original heuristic). On the contrary, as this value converges to 0, the selection behavior imitates a uniform-random one. Of course, intermediate values of β allow to consider randomization policies between both extremes, which makes the construction process more effective (De Armas et al. [66]).

Algorithm 1 illustrates this procedure. It starts by obtaining the job information from the instance (line 1). The main loop at lines 2 to 10 builds a solution by adding one job at a time at the last position of a partial solution until all jobs have been incorporated into the permutation.

Inside this loop, the index ξ'_{jk} is calculated for each job (lines 3 to 5). Then, the jobs are sorted in ascending order by this index value (line 6). The biased-randomized selection procedure is employed to select the next job to be added to the partial solution (line 7). Finally, the job selected is allocated at the last position of the partial solution (line 8) and it is removed from the list of unscheduled jobs (line 9).

Algorithm 1: BR-FF Algorithm

```
1 L ← {1, ..., n};
2 while L ≠ ∅ do
3 |   foreach j ∈ L do
4 |   |   Compute ξ'_{j,k} ;
5 |   Sort the jobs in the ascending order by ξ'_{j,k}
6 |   Choose, at biased randomization a job index j ∈ L
7 |   S ← S‖J_{j*};
8 |   L ← L \ {j*};
9 return S;
```

4.1.3 Extending the BR-FF Algorithm to a Metaheuristic

Aiming at exploring the solution space of the BR-FF detailed in Section 4.1.2, the metaheuristic variable neighborhood descent (VND) proposed by Hansen and Mladenovic [67], was incorporated into their framework. Due to their easy implementation, the aforesaid method was chosen (Hansen et al. [68]).

The VND method is composed of an initial solution generated by a constructive heuristic and local search operators. On this metaheuristic, the neighborhood of the initial solution is explored by the local search methods, i.e, small modifications on the initial solutions are made until a local optimum is reached. When it occurs, the modified solution replaces the original one and the method continues until a search strategy is reached. For more details about VND, readers are referred to Hansen and Mladenovic [67].

In this work, the proposed metaheuristic method is denoted as BR-FF-VND, and it is composed of the proposed BR-FF defined in Section 4.1.2 as a constructive heuristic and two classical local searches operators based on the work carried out by Den Besten and Stützle [69], insertion and interchange. The local search interchange aims to swap two jobs from the original positions, while the insertion method attempts to move a job to a new position that improves the solution benefit, maintaining the new order in case of the new solution found has a higher benefit than the original ones. The methods is perform until the search strategy first improvement is reached, i.e., the method is executed until a better solution than the solution provided by the constructive heuristic (BR-FF) is found.

4.2 Extending the BR-FF-VND Algorithm to a Simheuristic

Since the main goals of this research is to solve the stochastic version of the PFSPDP, this Section explains how Monte Carlo simulation (MCS) has been integrated into the BR-FF-VND algorithm in order to build a simheuristic. Considered as a special case of simulation-optimization (Juan et al. [7]), the simheuristic approach is not only able to deal with a stochastic version of most combinatorial optimization problems but also to provide risk / reliability analyses on the stochastic solutions it generates. Thus, simheuristic approaches have been gaining notoriety during the last years, being presented in different fields such as: flow-shop problems (Hatami et al. [9]; Gonzalez-Neira et al. [70]), finance (Panadero et al. [71]), vehicle routing problems (Guimarans et al. [72]; Calvet et al. [73]; Onggo et al. [74]; Reyes-Rubiano et al. [75]), arc routing problems (Gonzalez-Martin et al. [76]), inventory routing problems (Gruler et al. [77]; Onggo et al. [78]; Gruler et al. [79]; Raba et al. [80]), or facility–location problems (Pagès-Bernaus et al. [81]; De Armas et al. [66]). As a novelty with respect to these previous references on simheuristic algorithms, in this work risk analyses were performed based on two well-known metrics: the value at risk (VaR_α) and the conditional value at risk (CVaR_α). These metrics contribute to provide more complete information to decision makers.

In this stochastic variant of the PFSPDP, the processing times of job j in machine l are modeled as a random variable, P_{jl}, which follows a given probability distribution. Despite this methodology can use any probability distribution that fits the historical data, in the computational experiments performed in this work, it has been assumed that P_{jl} follows a log-Normal probability distribution with location parameter μ_{jl} and scale parameter σ_{jl} – actually, due to their flexibility, the log-Normal and Weibull probability distributions are frequently employed in reliability analysis to model positive failure times (Wolstenholme [82]). Other authors have also used the log-Normal distribution to model processing times (Baker and Altheimer [42]; Juan et al. [10]; Framinan and Perez-Gonzalez [57]). In order to extend the deterministic instances in a natural way, it has been assumed that $E[P_{jl}] = p_{jl}$. Likewise, in order to analyze different uncertainty levels, it has been assumed that $Var[P_{jl}] = h \cdot p_{jl}$, where $h \in \{0.1, 0.5, 1.0, 2.0\}$ is a design parameter. Equations 4-4 and 4-5 display, for the log-Normal distribution, the relationship between the location / scale parameters and its expected / variance values.

$$\mu_{jl} = lnE[P_{jl}] - \frac{1}{2}\left(1 + \frac{Var[P_{jl}]}{E[P_{jl}]^2}\right) \tag{4-4}$$

$$\sigma_{jl} = \left|\sqrt{\ln\left(1 + \frac{Var[P_{jl}]}{E[P_{jl}]^2}\right)}\right| \tag{4-5}$$

Aiming at complement this PFSPDPST environment a PFSPSTRD was proposed by taken into account uncertainties associated with the job j availability. This has occurred by the assumption of the release date rj of each job $j \in \mathcal{J}$ in the first machine $l = 1$ is not known in advance and follows log-normal probability distribution with location and scale parameters defined as it illustrated in Equations 4-6 and 4-7.

The log normal probability distribution was chosen to model the stochastic release dates since this probability distribution do not require complex parameters to model stochastic variables and it is a more natural choice to model non-negative random variables than the normal distribution (Juan et al. [83]).

Aiming to study different scenarios of uncertainty, it has been assumed that $Var[r_j] = h \cdot r_j$, where $h \in \{0.1, 0.5, 1.0, 2.0\}$ is a design parameter.

$$\mu_j = lnE[r_j] - \frac{1}{2}\left(1 + \frac{Var[r_j]}{E[r_j]^2}\right) \tag{4-6}$$

$$\sigma_j = \left|\sqrt{\ln\left(1 + \frac{Var[r_j]}{E[r_j]^2}\right)}\right| \tag{4-7}$$

Algorithm 2 portrays the simulation process.

Algorithm 2: Simulation Algorithm

```
1  vsolSim ← Solution from the deterministic environment
2  foreach Simulation ∈nRuns do
3      foreach j ∈ 𝒥 do
4          foreach l ∈ ℳ do
5              γ ← Deterministic variable
6              γ' ← Calculate μ_jl and σ_jl of γ following a log-normal
                probability distribution
7      SimSolution ← vsolSim(γ')
8      vReward ← SimSolution.Reward
9  Average reward ← vReward.Average
10 return Average reward
```

The algorithm starts getting the solution provided by the deterministic environment as it was previously discussed in Section 2.2 (line 1). For each run of the simulation, it is calculated for each job in each machine the average and standard deviations of the variable that will be considered as following a log-normal probability distribution (lines 3 to 6), as it was previously discussed in Section 4.2. Then the scheduling is calculated using the simulated parameters (lines 7)(it is important to highlight that the permutation of jobs still the same as informed in line 1). Finally, the reward is computed (line 8).

In the deterministic PFSPDP, the reward associated with a given permutation of jobs is calculated by the step-wise function provided in Equation 2-1. However, in the stochastic PFSPDP the reward associated with a given permutation is a random variable that will take a different value each time the solution is tested in a real-life or simulated environment. Hence, an estimate of the expected demand associated with a given permutation will be given by the average reward obtained after executing a sufficient number of simulation runs (nRuns) (line 9 and 10).

Algorithm 3 depicts the proposed Sim BR-FF-VND procedure.

Algorithm 3: Sim BR-FF-VND

```
1  initSol ← BR-FF
2  bestSol ← initSol
3  fastSimulation(bestSol)
4  insert(poolBestSol, bestSol)
5  baseSol ← bestSol
6  while Stopping criterion is not reached do
7      initNewSol ← BR-FF
8      newSol ← initNewSol
9      while initNewSol.getReward > newSol.getReward do
10         newSol ← localSearch.Interchange(initNewSol)
11         newSol ← localSearch.Insertion(newSol)
12     Δ ← newSol.getReward - baseSol.getReward
13     if Δ > 0 then
14         fastSimulation(newSol)
15         Δ′ ← newSol.getStochReward − baseSol.getStochReward
16         if Δ′ > 0 then
17             baseSol ← newSol
18             if newSol.getStochReward > bestSol.getStochReward then
19                 bestSol ← newSol
20                 insert(poolBestSol, bestSol)
21 for solution ∈ poolBestSol do
22     longSimulation(solution)
23     if solution.getStochReward > bestSol.getStochReward then
24         bestSol ← solution
25 return bestSol
```

The main ideas behind this algorithm are explained next. First, an initial solution for the deterministic PFSPDP is generated by employing the BR-FF (line 1). This is considered as the best solution so far (line 2). A fast simulation is performed to measure, in the stochastic environment, the reward provided by the best solution so far (line 3). Then, the resulting solution is inserted in the pool of best solutions (line 4). While the stopping criterion is not satisfied, a while loop is performed (line 6 to 25). This loop starts with the generation of a new solution in the deterministic environment (line 7). This solution is then enhanced by using the VND component (local search operators) (line 9 to 12). The Δ value checks the quality of this new solution in the deterministic

environment (line 13). If this new solution offers a higher reward than the current base solution (line 13), the new solution is considered as a good candidate for solving the deterministic version of the problem and a promising solution for the stochastic version. Notice that a correlation between the best deterministic and the best stochastic solutions is assumed in this approach. In order to estimate the quality of the new solution in the stochastic environment, a fast simulation is run (line 15). If this new solution offers a higher expected reward than the current base solution (i.e., $\Delta' > 0$) (line 17), the new solution is considered as a good candidate for solving the stochastic version and the base solution is updated (line 18). Moreover, if this new solution shows a higher expected reward than the current best solution (line 19), the latter is updated by the former (line 20), which is also inserted in the pool of best solutions (line 21). Finally, the pool of best solutions found are analyzed in more detail using an intensive simulation (i.e., employing a larger number of simulation runs) (line 26). This allows to increase the accuracy of the estimated expected rewards associated with each solution and generate a final sort of the best solutions based on this criterion (line 29).

In addition to providing accurate estimates of expected rewards, the output provided by the intensive simulation process can also be used to compute two well-known risk metrics: the value-at-risk (VaR_α) and the conditional value-at-risk (CVaR_α), which are going to be computed for $\alpha \in \{95\%, 97.5\%, 99\%\}$.

To compare the goodness of simheuristic results, the Algorithm 4 was proposed, in which the best solution provided by BR-FF-VND for the deterministic PFSPDP is analyzed in the stochastic environment by considering the processing times as random variables following a log-normal probability distribution with μ_{jl} and σ_{jl}.

Algorithm 4: Deterministic BR-FF-VND simulated

```
1  bestSol ← BR-FF
2  while Stopping criterion is not reached do
3      initNewSol ← BR-FF
4      newSol ← initNewSol
5      while initNewSol.getReward > newSol.getReward do
6          newSol ← localSearch.Interchange(initNewSol)
7          newSol ← localSearch.Insertion(newSol)
8      if newSol > BestSol then
9          bestSol ← newSol
10 longSimulation(bestSol)
11 return bestSol
```

The Algorithm 4 starts executing the proposed BR-FF heuristic, which is considered as the best solution so far (line 1). Then while the stop criterion is not satisfied the loop between lines 2 to 9 is performed. The loop starts with the execution of the BR-FF-VND (lines 3 to 7), and in case the new solution has a higher reward than the current best solution, the latter one is updated (lines 8 and 9). Finally, after the stop criterion is reached, a long simulation is executed to analyze the behavior of the best deterministic solution in the stochastic environment (line 10).

4.3 Risk Metrics

Generally associated with a loss distribution, the risk metrics, VaR_α and CVaR_α, were approached in this study in a revenue context. To this end, the definition provided by Street [84] has been followed: CVaR_α represents the conditional expected value of the revenue left-side worst distribution scenario, below a given $1-\alpha$ quantile, which is known as VaR_α. The latter is conventionally defined as a maximum loss in a given $1-\alpha$ quantile (Pflug [85]). In other words, in a revenue distribution for $\alpha = 95\%$, $\text{VaR}_{95\%}$ is the quantile of the 5% worst benefits. Meanwhile, $\text{CVaR}_{95\%}$ is the average of the 5% worst benefits. Despite both metrics aims at measuring the risk associated with a probability distribution, the CVaR_α developed by Rockafellar and Uryasev [86] is considered a better risk metric than the VaR_α, since the former is considered a coherent measure with the four axioms defined by Artzner et al. [87]: monotonicity, translation invariance, positive homogeneity, and subadditivity. Thus, CVaR_α is able to quantify events beyond VaR_α by representing the risks reflected at extreme tails of the probability distribution (Rockafellar and Uryasev [88]; Sarykalin et al. [89]).

For an overview of the applications of the Conditional Value at Risk (CVaR_α), readers are referred to Filippi et al. [90].

5
Results and Discussion

5.1
Experimental Settings

This Section describes the instances, the computational resources, and the design parameters employed in the computational experiments.

Aiming to solve this PFSPDP in a deterministic and stochastic context, the goodness of the solutions provided by the proposed methods defined in Section 4 were analyzed. To do it so, first, the PFSPDP was analyzed into the deterministic context, since the assumption of a strong correlation between the best deterministic and stochastic solutions was considered, as mentioned before in Section 2.2 and then PFSPDP was analyzed into a stochastic environment.

5.1.1
Benchmark Instances

In the computational experiments, the 150 instances introduced by Pessoa and Andrade [2] have been used. These instances, which are available at https://data.mendeley.com/datasets/m2wcd42pvy/1, were randomly generated and are composed of a set of $n = 100$ jobs that need to be processed by $m = 2$ machines. The deterministic processing times were randomly generated using an integer uniform distribution in the interval $[1, 100]$. The delivery dates were generated using the number k of delivery dates, with $k \in \{1, 2, 3, 5, 7, 10\}$. The makespan C_{max} was obtained as usual (Johnson [12]). The first delivery date is set to $d_1 = \gamma \cdot C_{max}/k$. The remaining delivery dates are set as $d_k = k \cdot d_1$, with $1 < k \leq K$ and $\gamma \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$.

Following Seddik et al. [1], release dates were randomly chosen in the interval $r_k = [d_{k-1}, d_{k-1} + r \cdot d_{k-1}]$, with $d_0 = 0$, $k \in \{1, 2, \ldots, K\}$, and $r \in \{0.1, 0.3, 0.5, 0.7, 1.0\}$. Instance names can be read as $k < k > a < \gamma > r < r >$. For example, instance $k10a1.0r0.1$ has 10 delivery dates, γ equals to 1.0 and r equals to 0.1

5.1.2 Computational Environment & Parameter Settings

The proposed algorithms have been implemented in the C++ programming language and executed in an Intel Core $i7-3960$ CPU at 3.30 GHz and 24.0 GB RAM running on a Windows operating system. A simple experiment was carried out in order to identify the best values for the β parameter employed during the biased-randomization process of the BR-FF algorithm. A total of 18 values of β were 10 times tested, with a time limit of 60 seconds for each test. Also, as suggested in Pessoa and Andrade [2], the following values were assigned to the FF heuristic parameters: $a = 4.0$ and $b = 0.0$. For each value of β, the percentage deviation between the proposed BR-FF heuristic and the FF heuristic was computed. The BR-FF$_{reward}$ has been considered as the average of the best solutions provided by a β, and FF$_{sol}$ as the average of the solutions provided by the FF heuristic. Hence, the relative percentage deviation is computed as: (BR-FF$_{reward}$− FF$_{sol}$)/ FF$_{sol}$. Note that a positive value of this percentage means that BR-FF$_{reward}$ is, on average, a better solution than FF$_{sol}$.

Figure 5.1 illustrates the deviation between the FF heuristic and the proposed BR-FF algorithm for different values of β. For a β value of 0.45 and higher the BR-FF is able to outperform the FF heuristic. The best solutions are obtained for $\beta = 0.95$. In other words, small changes in the job selection process, i.e., a small change in the logic of the original FF heuristic, allows better solutions to be found. For β values close to zero, the proposed heuristic assumes a random behavior, finding worst solutions than the original FF heuristic, being this behavior explained by the non-maintained of the logic of the original heuristic allowing jobs with higher values of $\xi'_{j,k}$ to be allocated in the first positions of the scheduling impacting the reward, since fewer jobs will finish sooner in the second machine.

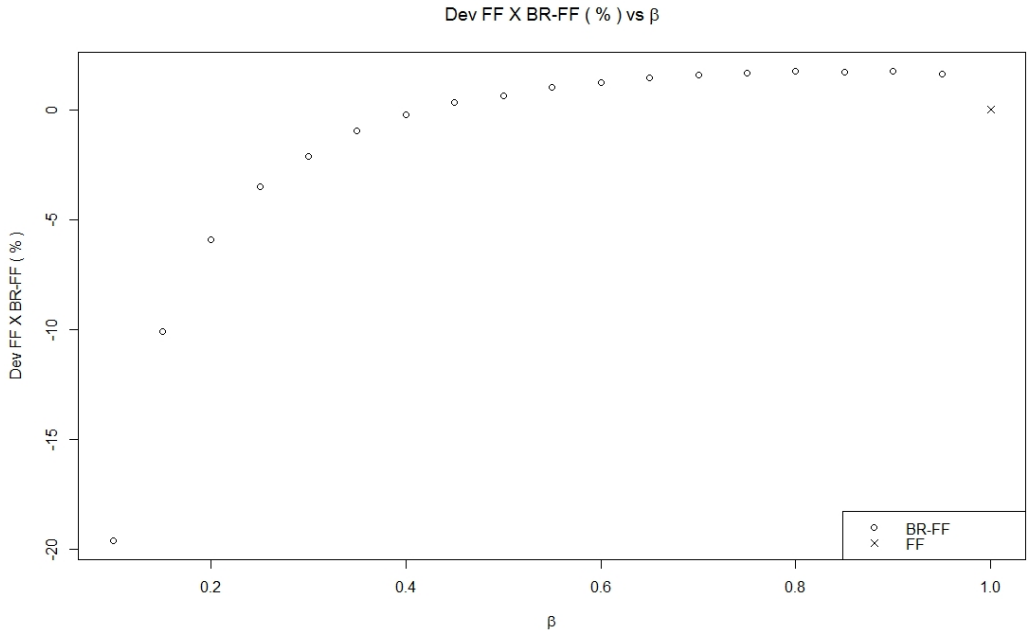

Figure 5.1: Deviation of best solutions between the BR-FF and FF heuristics versus the β parameter.

5.2 Testing the Biased-Randomized Algorithm in the Deterministic Scenario

According to the experiments carried out by Pessoa and Andrade [2], the BRKGA and their FF-based iterated local search (ILS) are among the best approaches for solving the deterministic PFSPDP. As explained in Grasas et al. [91], the ILS framework proposed by Lourenço [92] can be easily extended into a full simheuristic algorithm. For that reason, this Section compare the performance of the proposed BR-FF-VND algorithm against the performance of the FF heuristic (base solving method) and that of the FF-ILS metaheuristic.

All instances were 10 times run using each of the solving approaches, allowing up to one minute per instance and approach. As could be noticed in Table 5.1, the FF-ILS method proposed by Pessoa and Andrade [2] shown the worst average of best-solution values (166.17). The proposed methods, BR-FF and BR-FF-VND, report an average of 166.35 and 167.25, respectively (i.e., they improve the FF-ILS by 0.11% and 0.65%, respectively). In addition, a GRASP-based algorithm (Resende and Ribeiro [93]) has also been implemented from the FF constructive method. This FF-GRASP provides results that improve the ones obtained with the FF heuristic. However, these results are somewhat below the ones given by the FF-ILS algorithm.

Algorithm						Deviation (%)			
Time (min)	FF (1)	FF-GRASP (2)	FF-ILS (3)	BR-FF (4)	BR-FF-VND (5)	(2)-(1)	(3)-(1)	(4)-(1)	(5)-(1)
1	163.71	165.24	166.17	166.35	167.25	0.93	1.50	1.61	2.16

Table 5.1: A comparison of algorithms for 1 minute computational time.

An additional experiment was carried out to compare the solutions generated by each algorithm with the ones provided by the FF heuristic. For each instance, two indicators were computed: *(i)* the best percentage deviation, *BestDev(%)*; and *(ii)* the average percentage deviation, *AvgDev(%)*. For each instance, the former measures the percentage gap between the best-found solution using a new algorithm – out of the 10 runs executed per instance – and the solution provided by the FF heuristic. Similarly, the latter measures the percentage gap between the average solution value – computed from the 10 runs – and the solution value provided by the FF heuristic.

Figure 5.2 depicts a box-plot of both percentage deviations, *AvgDev(%)* and *BestDev(%)* across all the instances for 1 minute computational time. The *AvgDev(%)* box-plot shows that the FF-ILS has a similar expected value (2.24) than the BR-FF (2.09). The same can be observed for the *BestDev(%)*. The proposed BR-FF-VND shows the higher deviations of *AvgDev (%)* and *BestDev (%)*, with an average of 2.94 and 3.53, respectively. Moreover, this algorithm also shows the highest average of the best solutions (167.25) among the methods analyzed. When compared to the FF-ILS, the proposed BR-FF-VND provides higher rewards for 59.33% of the instances.

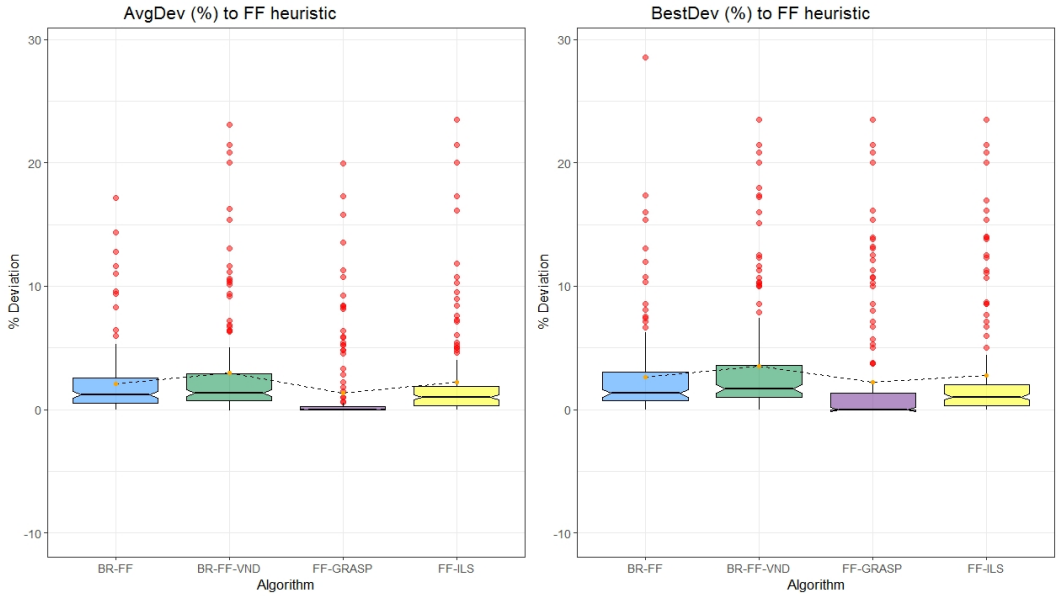

Figure 5.2: Box-plot of the Percentage deviation *AvgDev (%)* and BestDev (%).

In summary, according to the experiment results, the BR-FF-VND seems to be the best-performing approach among the ones tested for the deterministic PFSPDP. Hence, it seems to be a good candidate to be extended into a simheuristic.

With the purpose of analyzing the performance of the proposed methods when more computational time is available, new tests were run using 10 seeds. These tests used 5, 10, and 30 minutes, respectively. Table 5.2 shows the results. The first column indicates the run time, while the next five columns depict the average of best-found solutions for each method. The last four columns provide the percentage deviations between the average of the best solutions for each algorithm and the average of the solutions obtained with the FF heuristic.

Algorithm						Deviation (%)			
Time (min)	FF (1)	FF-GRASP (2)	FF-ILS (3)	BR-FF (4)	BR-FF-VND (5)	(2)-(1)	(3)-(1)	(4)-(1)	(5)-(1)
5	163.71	165.66	166.46	166.61	167.63	1.19	1.68	1.77	2.39
10	163.71	165.93	166.60	166.73	167.73	1.36	1.77	1.84	2.46
30	163.71	166.25	166.77	166.93	167.83	1.55	1.87	1.97	2.52

Table 5.2: A comparison of algorithms for different computing times.

Notice that, regardless the computational time employed, the BR-FF-VND algorithm shows a better average performance than the FF-ILS one. All the detailed results from this Section are available at https://doi.org/10.17771/PUCRio.ResearchData.48322.

5.3 Testing a Simheuristic in the Stochastic Scenario

As explained in Section 4.2, the aforementioned instances were generalized by using the log-Normal probability distribution to model processing times. Then, this stochastic version of the problem is solved by employing the Sim BR-FF-VND algorithm introduced in the same Section with 100 runs for fast simulation and 1.000 runs in a long simulation, considering different levels of uncertainty, i.e., $h \in \{0.1, 0.5, 1.0, 2.0\}$.

For each uncertainty level, Figure 5.3 displays the percentage deviation between the best solutions, *BestDev (%)*, found by the Sim BR-FF-VND (stochastic version) and the best solutions found by the BR-FF-VND (deterministic version). For all experiments performed in the Section, the 150 instances were run using 10 seeds and 1-minute computational time as simheuristic's stop criterion. All the detailed results are available at https://doi.org/10.17771/PUCRio.ResearchData.48322.

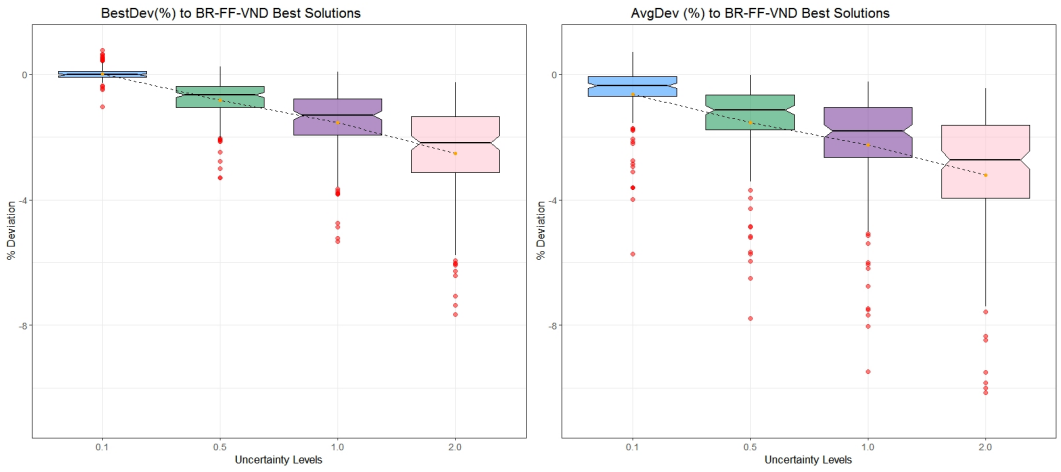

Figure 5.3: Percentage deviation of *BestDev*(%) and *AvgDev*(%) to BR-FF-VND in the deterministic environment.

Notice that the uncertainty level directly impacts on the average of the expected rewards. For a low level of uncertainty ($h = 0.1$), the average of *BestDev (%)* shows a value close to 0 (0.044 ± 0.30). On the contrary, as the uncertainty level increases the stochastic solutions diverge from the

deterministic ones. It is important to highlight that despite most of the instances analyzed increases in the processing times have occurred more often than decreases, the latter one was also considered in the simulation process. As could be noticed in Figure 5.3 for a lower level of uncertainty ($h = 0.1$) the simheuristic has found higher solutions than the BR-FF-VND (deterministic version) for a set of instances, this has occurred since more jobs have been processed faster, having finished earlier in the second machine. The relative percentage between the average of all the best stochastic solutions and the best deterministic solution, *AvgDev (%)*, was also analyzed, showing a similar behavior as the one already described for the *BestDev (%)*.

Another experiment was carried out to illustrate the goodness of the simheuristic results. To do it so, for each instance the best deterministic solution provided by the proposed deterministic BR-FF-VND was 1.000 times simulated using MCS as it was portrayed in Algorithm 4.

Then, as illustrated in Figure 5.4 the relative percentage deviation between the best solutions, *BestDev (%)*, found by the Sim BR-FF-VND and the simulated deterministic solution was calculated across all the instances for different levels of uncertainty.

As it was stated in Juan et al. [10], the simheuristic approach can find solutions between a lower and upper bound delimited by the deterministic solutions. This behavior was noticed for the average of a simheuristic results since it has found on average better solutions than the simulated deterministic solutions as it illustrated in Figure 5.4, which are considered as lower bound, and worst solutions on average than the best deterministic solutions as it illustrated in Figure 5.3, which are considered as upper bound.

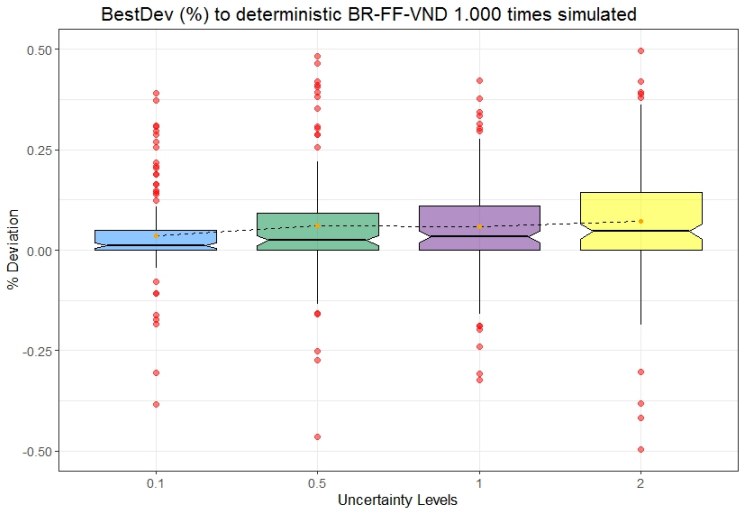

Figure 5.4: Percentage deviation of the best solutions - $BestDev(\%)$ between the Sim BR-FF-VND and the deterministic BR-FF-VND 1.000 times simulated.

Since risk analysis have an important role on the quantification of the worst possible scenarios, the proposed simheuristic approach has been expanded to consider a full risk analysis via the quantification of the value at risk VaR_α and CVaR_α. For instance *k10a0.3r0.1* and uncertainty level $h = 1.0$, Figure 5.5 portrays the VaR_α and $CVaR_\alpha$ values for three differences levels of α (95%, 97.5%, and 99%). This instance has an expected reward of 250.44. The VaR_α could be interpreted as the worst scenario of a reward for a level of confidence α. In this case, the associated rewards would be 241, 239, and 235, respectively. Likewise, the CVaR_α can be interpreted as the average of the worst rewards for a level of confidence α. In this instance, the associated values are 238.10, 235.76, and 233.30, respectively.

Figure 5.6 depicts the behavior that has been observed across all the instances for the different values of α. Notice that as the level of uncertainty is increased, it has an impact on the processing times variability. This, in turn, influences on the expected rewards generated, since more jobs will finish latter on the last machine. Hence, the values of both risk metrics are reduced as the level of uncertainty increases.

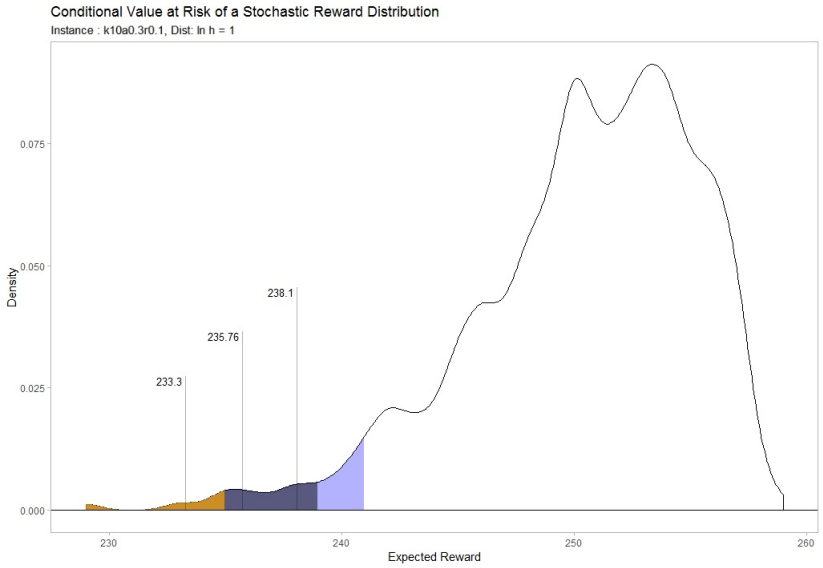

Figure 5.5: Conditional VaR for instance *k10a0.3r0.1* using different levels of α and $h = 1.0$.

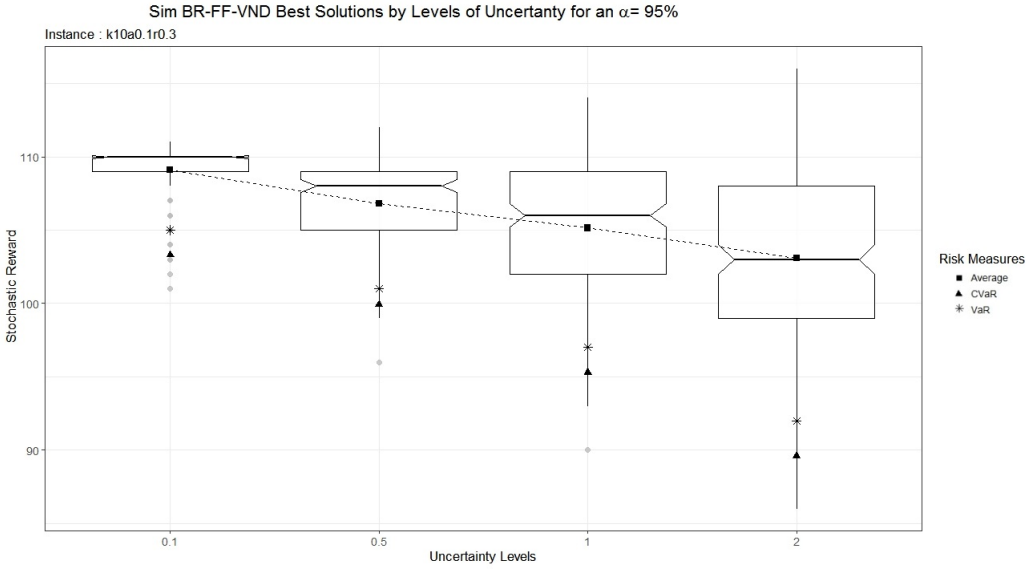

Figure 5.6: Behavior of the $\text{VaR}_{95\%}$ and $\text{CVaR}_{95\%}$ on different levels of uncertainty for instance *k10a0.1r0.3*.

As aforesaid in Section 2.2, an extension of the PSFPST into a more realistic environment, which considers the uncertainty in the availability of the job, is also proposed.

In this stochastic environment named PSFPSTRD, four different levels of uncertainty $h \in \{0.1, 0.5, 1.0, 2.0\}$ have been considered to investigate the impact of those in the expected rewards calculated by the step-wise function defined in Section 2.1.

Figure 5.7 portrays the percentage deviation between the best solutions, *BestDev (%)*, and the average of all the best solutions AvgDev (%), found by the Sim BR-FF-VND (stochastic version) and the best solutions found by the BR-FF-VND (deterministic version) on this PSFPSTRD environment for each uncertainty scenario. The same behavior observed in the PFSPDPST analysis was noticed for both deviations, i.e., as the level of uncertainty increase, the average of the expected rewards decreases. Moreover, as it was expected and as it was previously discussed in the PSFPST analyses, the risk metrics the VaR_α and CVaR_α are directly impacted by the level of uncertainty, as it could be noticed in Figure 5.8, as the level of uncertainty, increases lower are the worst scenarios of expected rewards.

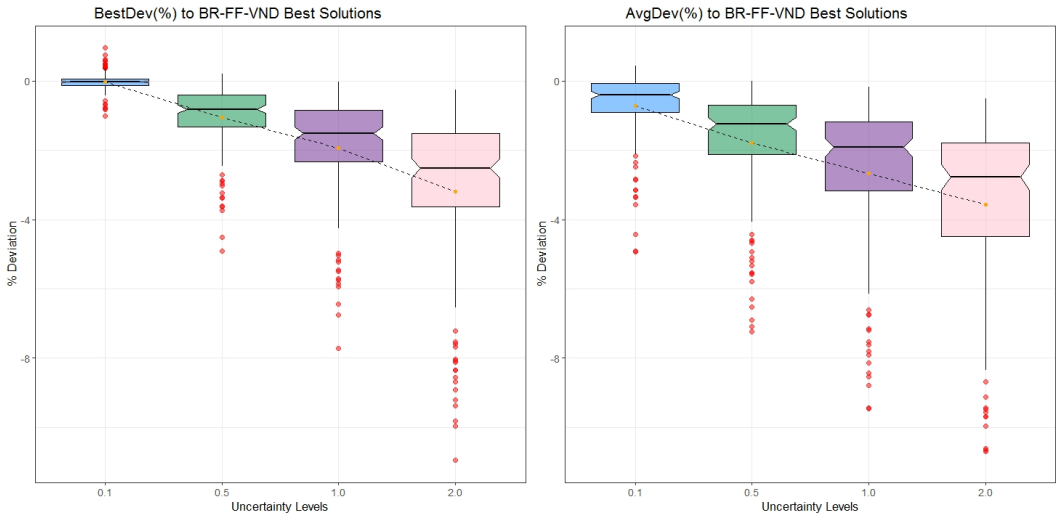

Figure 5.7: Percentage deviation of *BestDev* (%) and *AvgDev* (%) to BR-FF-VND in the deterministic environment.

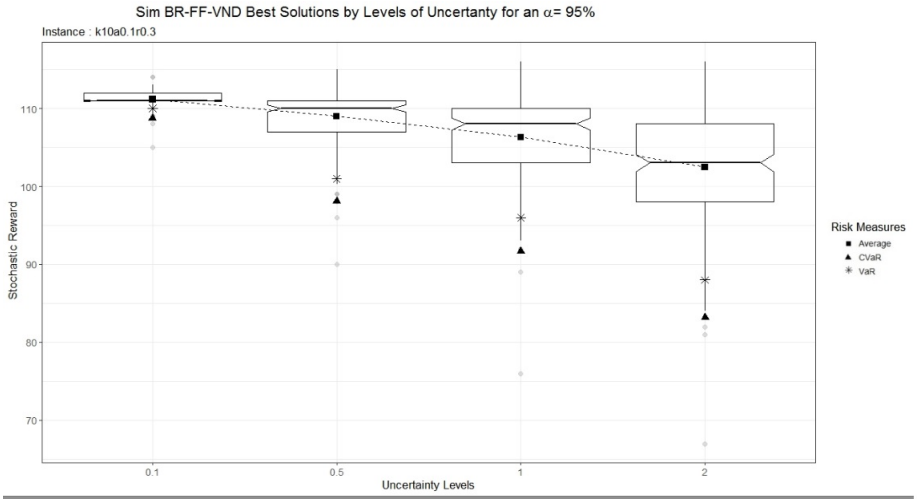

Figure 5.8: Behavior of the $VaR_{95\%}$ and $CVaR_{95\%}$ on different levels of uncertainty for instance *k10a0.1r0.3* in a PSFPSTRD

Regarding the simheuristic best solutions in a PFSPSTRD and in the PFSPDPST environment, a experiment was carried out to computed their deviations.

As illustrated in Figure 5.9, the simheuristic in the PFSPSTRD has found on average worst solutions than the simheuristic best solutions in a PFSPDPST environment in all the levels of uncertainty analysed. This behavior portraits that not only variations in the processing times impact the expected rewards but also the variation the release dates, since the job is more often available to be process in the first machine latter and consequently less job finished earlier in the last machines.

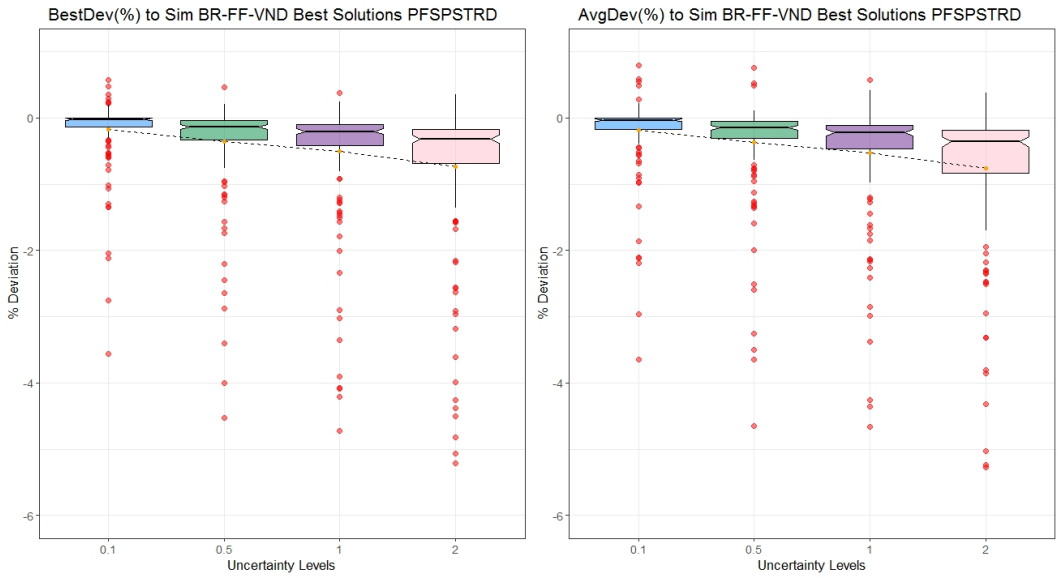

Figure 5.9: Percentage deviation of *BestDev*(%) and *AvgDev*(%) to SimBR-FF-VND in PFSPSTRD

By all of the analyses performed, it is possible to conclude that regardless of the uncertainty level, scenarios lower rewards are expected when compared to the deterministic version of the problem. Hence the decision-maker when considering the uncertainty in this PFSP should be cautious with the expected revenue associated with the process and depending on his risk-profile should do preventive measures in the company more softly or aggressively aiming at having a profit.

6
Conclusions

This work analyzes a stochastic version of the Permutation Flowshop Scheduling Problem with Delivery Dates and Cumulative Payoffs (PFSPDP).This problem is motivated by a real-life case regarding the books digitization in a library. In order to do so, an efficient biased-randomized algorithm was develop for the deterministic version, and then this algorithm was extended into a simheuristic by integrating Monte Carlo simulation into the metaheuristic framework. Hence, the FF heuristic proposed by Fernandez-Viagas and Framinan[4] is first extended into a biased-randomized algorithm (BR-FF). Next, a variable neighborhood descent (VND) metaheuristic framework is incorporated to the BR-FF. Finally, the BR-FF-VND is extended into a simheuristic algorithm that can solve the PFSPDP with stochastic processing times and PFSPDP with stochastic processing times and release dates.

An extensive set of experiments was carried out to test the quality of the proposed algorithms. The proposed solutions for the deterministic version of the problem shown to be a promising method when compared with the recently published work by Pessoa and Andrade [2] since it have found better solutions than the FF-ILS proposed by Pessoa and Andrade [2] in all the computational times analyzed in this work. Moreover, the propose simheuristic approach is the first one in the literature dealing with the stochastic versions of the PFSPDP. From the experiments, it was observed that different levels of uncertainty have a direct impact on the stochastic reward. Also, the worst-case scenarios were analyzed by using two well-known risk metrics: the value-at-risk (VaR_α) and the conditional-value-at-risk (CVaR_α). As expected, an increase in the level of uncertainty has an impact on the tails of the probability distribution used to model stochastic processing times. This pattern affects the rewards computed by the VaR_α and CVaR_α, since both risk metrics consider the behavior of the rewards in the tails of the distribution.

By the experiment and analysis carried out in this work, the combination of stochastic optimization and risk measures analyses has shown to be a promising combination to help decision-makers.

Since the analysis of different stochastic scenarios of rewards allows preventive measures to be taken, such as the analyses of cost and expenses, cash

flow, marketing position, short and long term strategic plan, etc, to diminish the impact of lower revenue in the company.

Several research lines can be considered for future work, among them: *(i)* to consider a bi-objective formulation of the problem, where both expected reward and solution risk are simultaneously optimized ;*(ii)* to incorporate other sources of randomness in the PFSPDP, such as machine breakdowns; *(iii)* to analyze the performance of the proposed biased-randomized algorithm when more complex functions are employed to model rewards by meeting deadlines; *(iv)* to analyze this problem in others flow-shop environment considering, for instance, more than two machines or considering as a parallel machine problem and *(v)* use other metaheuristics as ILS or GRASP, to explore the neighborhood of the solution provided by the BR-FF heuristic.

Bibliography

[1] SEDDIK, Y.; GONZALES, C. ; KEDAD-SIDHOUM, S.. **Single machine scheduling with delivery dates and cumulative payoffs**. Journal of Scheduling, 16(3):313–329, 2013.

[2] PESSOA, L. S.; ANDRADE, C. E.. **Heuristics for a flowshop scheduling problem with stepwise job objective function**. European Journal of Operational Research, 266(3):950–962, 2018.

[3] GRASAS, A.; JUAN, A. A.; FAULIN, J.; DE ARMAS, J. ; RAMALHINHO, H.. **Biased randomization of heuristics using skewed probability distributions: a survey and some applications**. Computers & Industrial Engineering, 110:216–228, 2017.

[4] FERNANDEZ-VIAGAS, V.; FRAMINAN, J. M.. **A new set of high-performing heuristics to minimise flowtime in permutation flowshops**. Computers & Operations Research, 53:68–80, 2015.

[5] FERRER, A.; GUIMARANS, D.; RAMALHINHO, H. ; JUAN, A. A.. **A BRILS metaheuristic for non-smooth flow-shop problems with failure-risk costs**. Expert Systems with Applications, 44:177–186, 2016.

[6] FERONE, D.; HATAMI, S.; GONZÁLEZ-NEIRA, E. M.; JUAN, A. A. ; FESTA, P.. **A biased-randomized iterated local search for the distributed assembly permutation flow-shop problem**. International Transactions in Operational Research, 27(3):1368–1391, 2019.

[7] JUAN, A. A.; KELTON, W. D.; CURRIE, C. S. ; FAULIN, J.. **Simheuristics applications: dealing with uncertainty in logistics, transportation, and other supply chain areas**. In: PROCEEDINGS OF THE 2018 WINTER SIMULATION CONFERENCE, p. 3048–3059. IEEE Press, 2018.

[8] GONZALEZ-NEIRA, E. M.; FERONE, D.; HATAMI, S. ; JUAN, A. A.. **A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times**. Simulation Modelling Practice and Theory, 79:23–36, 2017.

[9] HATAMI, S.; CALVET, L.; FERNÁNDEZ-VIAGAS, V.; FRAMIÑÁN, J. M. ; JUAN, A. A.. **A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem**. Simulation Modelling Practice and Theory, 86:55–71, 2018.

[10] JUAN, A. A.; BARRIOS, B. B.; VALLADA, E.; RIERA, D. ; JORBA, J.. **A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times**. Simulation Modelling Practice and Theory, 46:101–117, 2014.

[11] GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K. ; KAN, A. R.. **Optimization and approximation in deterministic sequencing and scheduling: a survey**. In: ANNALS OF DISCRETE MATHEMATICS, volumen 5, p. 287–326. Elsevier, 1979.

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

[13] FRAMINAN, J. M.; GUPTA, J. N. ; LEISTEN, R.. **A review and classification of heuristics for permutation flow-shop scheduling with makespan objective**. Journal of the Operational Research Society, 55(12):1243–1255, 2004.

[14] FERNANDEZ-VIAGAS, V.; RUIZ, R. ; FRAMINAN, J. M.. **A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation**. European Journal of Operational Research, 257(3):707–721, 2017.

[15] GUPTA, J. N.; STAFFORD JR, E. F.. **Flowshop scheduling research after five decades**. European Journal of Operational Research, 169(3):699–711, 2006.

[16] FRAMINAN, J. M.; LEISTEN, R.. **Total tardiness minimization in permutation flow shops: a simple approach based on a variable greedy algorithm**. International Journal of Production Research, 46(22):6479–6498, 2008.

[17] MLADENOVIĆ, N.; HANSEN, P.. **Variable neighborhood search**. Computers & operations research, 24(11):1097–1100, 1997.

[18] LIU, J.; REEVES, C. R.. **Constructive and composite heuristic solutions to the $p//\sum$ ci scheduling problem**. European Journal of Operational Research, 132(2):439–452, 2001.

[19] FRAMINAN, J.; LEISTEN, R.. **An efficient constructive heuristic for flowtime minimisation in permutation flow shops**. Omega, 31(4):311–317, 2003.

[20] FRAMINAN, J. M.; LEISTEN, R. ; RUIZ-USANO, R.. **Comparison of heuristics for flowtime minimisation in permutation flowshops**. Computers & Operations Research, 32(5):1237–1254, 2005.

[21] PASUPATHY, T.; RAJENDRAN, C. ; SURESH, R.. **A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs**. The International Journal of Advanced Manufacturing Technology, 27(7-8):804–815, 2006.

[22] NAGANO, M. S.; MOCCELLIN, J.. **Reducing mean flow time in permutation flow shop**. Journal of the Operational Research Society, 59(7):939–945, 2008.

[23] NAWAZ, M.; ENSCORE JR, E. E. ; HAM, I.. **A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem**. Omega, 11(1):91–95, 1983.

[24] DONG, X.; HUANG, H. ; CHEN, P.. **An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion**. Computers & Operations Research, 36(5):1664–1669, 2009.

[25] TASGETIREN, M. F.; PAN, Q.-K.; SUGANTHAN, P. N. ; CHEN, A. H.. **A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops**. Information sciences, 181(16):3459–3475, 2011.

[26] PAN, Q.-K.; TASGETIREN, M. F. ; LIANG, Y.-C.. **A discrete differential evolution algorithm for the permutation flowshop scheduling problem**. Computers & Industrial Engineering, 55(4):795–816, 2008.

[27] DELLA CROCE, F.; GROSSO, A. ; SALASSA, F.. **A matheuristic approach for the total completion time two-machines permutation flow shop problem**. In: PROCEEDINGS OF THE EUROPEAN CONFERENCE ON EVOLUTIONARY COMPUTATION IN COMBINATORIAL OPTIMIZATION, p. 38–47. Springer, 2011.

[28] PAN, Q.-K.; RUIZ, R.. **A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime**. Computers & Operations Research, 40(1):117–128, 2013.

[29] LEE, W.-C.; CHUNG, Y.-H.. **Permutation flowshop scheduling to minimize the total tardiness with learning effects**. International Journal of Production Economics, 141(1):327–334, 2013.

[30] MOLINA-SÁNCHEZ, L.; GONZÁLEZ-NEIRA, E.. **GRASP to minimize total weighted tardiness in a permutation flow shop environment**. International Journal of Industrial Engineering Computations, 7(1):161–176, 2016.

[31] ABEDINNIA, H.; GLOCK, C. H. ; BRILL, A.. **New simple constructive heuristic algorithms for minimizing total flow-time in the permutation flowshop scheduling problem**. Computers & Operations Research, 74:165–174, 2016.

[32] LAHA, D.; SARIN, S. C.. **A heuristic to minimize total flow time in permutation flow shop**. Omega, 37(3):734–739, 2009.

[33] YU, A. J.; SEIF, J.. **Minimizing tardiness and maintenance costs in flow shop scheduling by a lower-bound-based ga**. Computers & Industrial Engineering, 97:26–40, 2016.

[34] ROSSI, F. L.; NAGANO, M. S. ; SAGAWA, J. K.. **An effective constructive heuristic for permutation flow shop scheduling problem with total flow time criterion**. The International Journal of Advanced Manufacturing Technology, 90(1-4):93–107, 2017.

[35] FERNANDEZ-VIAGAS, V.; FRAMINAN, J. M.. **A beam-search-based constructive heuristic for the pfsp to minimise total flowtime**. Computers & Operations Research, 81:167–177, 2017.

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

[37] ROSSI, F. L.; NAGANO, M. S.. **Heuristics for the mixed no-idle flowshop with sequence-dependent setup times and total flowtime criterion**. Expert Systems with Applications, 125:40–54, 2019.

[38] TA, Q. C.; BILLAUT, J.-C. ; BOUQUARD, J.-L.. **Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem**. Journal of Intelligent Manufacturing, 29(3):617–628, 2018.

[39] ANDRADE, C. E.; SILVA, T. ; PESSOA, L. S.. **Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm**. Expert Systems with Applications, 128:67–80, 2019.

[40] ELYASI, A.; SALMASI, N.. **Stochastic scheduling with minimizing the number of tardy jobs using chance constrained programming**. Mathematical and Computer Modelling, 57(5-6):1154–1164, 2013.

[41] GONZALEZ-NEIRA, E. M.; MONTOYA-TORRES, J. R. ; CABALLERO-VILLALOBOS, J. P.. **A comparison of dispatching rules hybridised with monte carlo simulation in stochastic permutation flow shop problem**. Journal of Simulation, 13(2):128–137, 2019.

[42] BAKER, K. R.; ALTHEIMER, D.. **Heuristic solution methods for the stochastic flow shop problem**. European Journal of Operational Research, 216(1):172–177, 2012.

[43] BANERJEE, B.. **Single facility sequencing with random execution times**. Operations Research, 13(3):358–364, 1965.

[44] MAKINO, T.. **On a scheduling problem**. Operations Research Society of Japan, 8:32–44, 1965.

[45] TALWAR, P.. **A note on sequencing problems with uncertain job times**. Journal of the Operations Research Society of Japan, 9(3-4):93–97, 1967.

[46] CUNNINGHAM, A. A.; DUTTA, S. K.. **Scheduling jobs, with exponentially distributed processing times, on two machines of a flow shop**. Naval Research Logistics Quarterly, 20(1):69–81, 1973.

[47] MITTAL, B.; BAGGA, P.. **A priority problem in sequencing with stochastic service times**. Operations Research, 14:19–28, 1977.

[48] DODIN, B.. **Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops**. Computers & Operations Research, 23(9):829–843, 1996.

[49] KAMBUROWSKI, J.. **Stochastically minimizing the makespan in two-machine flow shops without blocking**. European Journal of Operational Research, 112(2):304–309, 1999.

[50] KAMBUROWSKI, J.. **On three-machine flow shops with random job processing times**. European Journal of Operational Research, 125(2):440–448, 2000.

[51] GOURGAND, M.; GRANGEON, N. ; NORRE, S.. **A review of the static stochastic flow-shop scheduling problem**. Journal of decision systems, 9(2):1–31, 2000.

[52] WANG, L.; ZHANG, L. ; ZHENG, D.-Z.. **A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing time**. The International Journal of Advanced Manufacturing Technology, 25(11-12):1157–1163, 2005.

[53] KALCZYNSKI, P. J.; KAMBUROWSKI, J.. **A heuristic for minimizing the expected makespan in two-machine flow shops with consistent coefficients of variation**. European Journal of Operational Research, 169(3):742–750, 2006.

[54] BAKER, K. R.; TRIETSCH, D.. **Three heuristic procedures for the stochastic, two-machine flow shop problem**. Journal of Scheduling, 14(5):445–454, 2011.

[55] CHOI, S.; WANG, K.. **Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach**. Computers & Industrial Engineering, 63(2):362–373, 2012.

[56] LIEFOOGHE, A.; BASSEUR, M.; HUMEAU, J.; JOURDAN, L. ; TALBI, E.-G.. **On optimizing a bi-objective flowshop scheduling problem in an uncertain environment**. Computers & Mathematics with Applications, 64(12):3747–3762, 2012.

[57] FRAMINAN, J. M.; PEREZ-GONZALEZ, P.. **On heuristic solutions for the stochastic flowshop scheduling problem**. European Journal of Operational Research, 246(2):413–420, 2015.

[58] GONZALEZ-NEIRA, E.; MONTOYA-TORRES, J. ; BARRERA, D.. **Flow-shop scheduling problem under uncertainties: Review and trends**. International Journal of Industrial Engineering Computations, 8(4):399–426, 2017.

[59] GHOLAMI-ZANJANI, S. M.; HAKIMIFAR, M.; NAZEMI, N. ; JOLAI, F.. **Robust and fuzzy optimisation models for a flow shop scheduling problem with sequence dependent setup times: A real case study on a PCB assembly company**. International Journal of Computer Integrated Manufacturing, 30(6):552–563, 2017.

[60] CUI, W.; LU, Z.; LI, C. ; HAN, X.. **A proactive approach to solve integrated production scheduling and maintenance planning problem in flow shops**. Computers & Industrial Engineering, 115:342–353, 2018.

[61] FRAMINAN, J. M.; FERNANDEZ-VIAGAS, V. ; PEREZ-GONZALEZ, P.. **Using real-time information to reschedule jobs in a flowshop with variable processing times**. Computers & Industrial Engineering, 129:113–125, 2019.

[62] POTTS, C.. **Analysis of heuristics for two-machine flow-shop sequencing subject to release dates**. Mathematics of Operations research, 10(4):576–584, 1985.

[63] LADHARI, T.; RAKROUKI, M. A.. **Heuristics and lower bounds for minimizing the total completion time in a two-machine flowshop**. International Journal of Production Economics, 122(2):678–691, 2009.

[64] FAULIN, J.; GILIBERT, M.; JUAN, A. A.; VILAJOSANA, X. ; RUIZ, R.. **SR-1: A simulation-based algorithm for the capacitated vehicle routing problem**. In: 2008 WINTER SIMULATION CONFERENCE, p. 2708–2716. IEEE, 2008.

[65] JUAN, A. A.; FAULIN, J.; RUIZ, R.; BARRIOS, B.; GILIBERT, M. ; VILAJOSANA, X.. **Using oriented random search to provide a set of alternative solutions to the capacitated vehicle routing problem**. In: OPERATIONS RESEARCH AND CYBER-INFRASTRUCTURE, p. 331–345. Springer, 2009.

[66] DE ARMAS, J.; KEENAN, P.; JUAN, A. A. ; MCGARRAGHY, S.. **Solving large-scale time capacitated arc routing problems: from real-time heuristics to metaheuristics**. Annals of Operations Research, 273(1-2):135–162, 2019.

[67] HANSEN, P.; MLADENOVIĆ, N.. **Variable neighborhood search**. In: Handbook of Metaheuristics, p. 145–184. Springer, 2003.

[68] HANSEN, P.; MLADENOVIĆ, N. ; PEREZ-BRITOS, D.. **Variable neighborhood decomposition search**. Journal of Heuristics, 7(4):335–350, 2001.

[69] DEN BESTEN, M.; STÜTZLE, T.. **Neighborhoods revisited: An experimental investigation into the effectiveness of variable neigh-**

borhood descent for scheduling. In: MIC'2001–4TH METAHEURISTICS INTERNATIONAL CONFERENCE, p. 545–549, 2001.

[70] GONZALEZ-NEIRA, E. M.; URREGO-TORRES, A. M.; CRUZ-RIVEROS, A. M.; HENAO-GARCÍA, C.; MONTOYA-TORRES, J. R.; MOLINA-SÁNCHEZ, L. P. ; JIMÉNEZ, J.-F.. **Robust solutions in multi-objective stochastic permutation flow shop problem**. Computers & Industrial Engineering, 137:106026, 2019.

[71] PANADERO, J.; DOERING, J.; KIZYS, R.; JUAN, A. A. ; FITO, A.. **A variable neighborhood search simheuristic for project portfolio selection under uncertainty**. Journal of Heuristics, p. 1–23, 2018.

[72] GUIMARANS, D.; DOMINGUEZ, O.; PANADERO, J. ; JUAN, A. A.. **A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times**. Simulation Modelling Practice and Theory, 89:1–14, 2018.

[73] CALVET, L.; WANG, D.; JUAN, A. ; BOVÉ, L.. **Solving the multidepot vehicle routing problem with limited depot capacity and stochastic demands**. International Transactions in Operational Research, 26(2):458–484, 2019.

[74] ONGGO, B. S.; PANADERO, J.; CORLU, C. G. ; JUAN, A. A.. **Agri-food supply chains with stochastic demands: A multi-period inventory routing problem with perishable products**. Simulation Modelling Practice and Theory, 97:101970, 2019.

[75] REYES-RUBIANO, L.; FERONE, D.; JUAN, A. A. ; FAULIN, J.. **A simheuristic for routing electric vehicles with limited driving ranges and stochastic travel times**. SORT-Statistics and Operations Research Transactions, 1(1):3–24, 2019.

[76] GONZALEZ-MARTIN, S.; JUAN, A. A.; RIERA, D.; ELIZONDO, M. G. ; RAMOS, J. J.. **A simheuristic algorithm for solving the arc routing problem with stochastic demands**. Journal of Simulation, 12(1):53–66, 2018.

[77] GRULER, A.; PANADERO, J.; DE ARMAS, J.; PÉREZ, J. A. M. ; JUAN, A. A.. **A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands**. International Transactions in Operational Research, 27(1):314–335, 2020.

[78] ONGGO, B. S.; JUAN, A. A.; PANADERO, J.; CORLU, C. G. ; AGUSTIN, A.. **An inventory-routing problem with stochastic demand and stock-out: A solution and risk analysis using simheuristics**. In: 2019 WINTER SIMULATION CONFERENCE (WSC), p. 1977–1988. IEEE, 2019.

[79] GRULER, A.; PANADERO, J.; DE ARMAS, J.; PÉREZ, J. A. M. ; JUAN, A. A.. **A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands**. International Transactions in Operational Research, 27(1):314–335, 2020.

[80] RABA, D.; ESTRADA-MORENO, A.; PANADERO, J. ; JUAN, A. A.. **A reactive simheuristic using online data for a real-life inventory routing problem with stochastic demands**. International Transactions in Operational Research, 2020.

[81] PAGÈS-BERNAUS, A.; RAMALHINHO, H.; JUAN, A. A. ; CALVET, L.. **Designing e-commerce supply chains: a stochastic facility–location approach**. International Transactions in Operational Research, 26(2):507–528, 2019.

[82] WOLSTENHOLME, L. C.. **Reliability modelling: a statistical approach**. Routledge, 2018.

[83] JUAN, A.; FAULIN, J.; GRASMAN, S.; RIERA, D.; MARULL, J. ; MENDEZ, C.. **Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands**. Transportation Research Part C: Emerging Technologies, 19(5):751–765, 2011.

[84] STREET, A.. **On the conditional value-at-risk probability-dependent utility function**. Theory and Decision, 68(1-2):49–68, 2010.

[85] PFLUG, G. C.. **Some remarks on the value-at-risk and the conditional value-at-risk**. In: PROBABILISTIC CONSTRAINED OPTIMIZATION, p. 272–281. Springer, 2000.

[86] ROCKAFELLAR, R. T.; URYASEV, S.. **Optimization of conditional value-at-risk**. Journal of risk, 2:21–42, 2000.

[87] ARTZNER, P.; DELBAEN, F.; EBER, J.-M. ; HEATH, D.. **Coherent measures of risk**. Mathematical finance, 9(3):203–228, 1999.

[88] ROCKAFELLAR, R. T.; URYASEV, S.. **Conditional value-at-risk for general loss distributions**. Journal of banking & finance, 26(7):1443–1471, 2002.

[89] SARYKALIN, S.; SERRAINO, G. ; URYASEV, S.. **Value-at-risk vs. conditional value-at-risk in risk management and optimization**. In: STATE-OF-THE-ART DECISION-MAKING TOOLS IN THE INFORMATION-INTENSIVE AGE, p. 270–294. Informs, 2008.

[90] FILIPPI, C.; GUASTAROBA, G. ; SPERANZA, M.. **Conditional value-at-risk beyond finance: a survey**. International Transactions in Operational Research, 27(3):1277–1319, 2020.

[91] GRASAS, A.; JUAN, A. A. ; LOURENÇO, H. R.. **SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization**. Journal of Simulation, 10(1):69–77, 2016.

[92] LOURENÇO, H. R.. **Job-shop scheduling: Computational study of local search and large-step optimization methods**. European Journal of Operational Research, 83(2):347–364, 1995.

[93] RESENDE, M. G.; RIBEIRO, C. C.. **Optimization by GRASP**. Springer, 2016.