Pontifícia Universidade Católica
do Rio de Janeiro

Victor Abu-Marrul Carneiro da Cunha

Solving the Deterministic and Stochastic Pipe-Laying Support Vessel Scheduling Problem

Tese de Doutorado

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

Advisor : Prof. Rafael Martinelli Pinto
Co-advisor: Prof. Silvio Hamacher

Rio de Janeiro
May 2021

Pontifícia Universidade Católica
do Rio de Janeiro

Victor Abu-Marrul Carneiro da Cunha

Solving the Deterministic and Stochastic Pipe-Laying Support Vessel Scheduling Problem

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

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

Prof. Silvio Hamacher
Co-advisor
Departamento de Engenharia Industrial – PUC-Rio

Prof. Irina Gribkovskaia
HiMolde

Prof. Virgílio José Martins Ferreira Filho
UFRJ

Prof. Bruno Fânzeres dos Santos
Departamento de Engenharia Industrial – PUC-Rio

Prof. Anand Subramanian
UFPB

Rio de Janeiro, May 03, 2021

Victor Abu-Marrul Carneiro da Cunha

Graduated in Industrial Engineering at the Veiga de Almeida University of Rio de Janeiro – UVA in 2013 and obtained his M.Sc. Degree in industrial Engineering at the Pontifical Catholic University of Rio de Janeiro in 2017.

Bibliographic data

Cunha, Victor Abu-Marrul Carneiro da

Solving the Deterministic and Stochastic Pipe-Laying Support Vessel Scheduling Problem / Victor Abu-Marrul Carneiro da Cunha ; advisor: Rafael Martinelli Pinto ; co-advisor: Silvio Hamacher. – 2021.

154 f. : il. color. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Industrial, 2021.

Inclui bibliografia

1. Engenharia Industrial – Teses. 2. Logística offshore. 3. Programação de embarcações. 4. Formulação matemática. 5. Métodos híbridos. 6. Metaheurísticas. I. Pinto, Rafael Martinelli. II. Hamacher, Silvio. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Industrial. IV. Título.

CDD: 658.5

Acknowledgments

To my family for all the support on this journey.
To my wife, Aline, for believing and encouraging me to go to the research area.
To my advisors for all their patience and collaboration during the last years.
To all professors and employees of the industrial engineering department at PUC-Rio, for all their assistance.
To my colleagues in the industrial department and Tecgraf institute for all the collaboration.
To CAPES, CNPq, Tecgraf Institute, and PUC-Rio for grants and support. Without these, none of this would be possible.
This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Cunha, Victor Abu-Marrul Carneiro da; Pinto, Rafael Martinelli (Advisor); Hamacher, Silvio (Co-Advisor). **Solving the Deterministic and Stochastic Pipe-Laying Support Vessel Scheduling Problem**. Rio de Janeiro, 2021. 154p. Tese de Doutorado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Offshore oil and gas exploration companies frequently need to deal with problems related to the efficient use of their resources. In this work, we address a ship scheduling problem associated with offshore oil and gas logistics – The Pipe Laying Support Vessel Scheduling Problem (PLSVSP). These vessels are specially designed to perform pipeline connections between sub-sea oil wells and production platforms. The connections are the last step to be performed to allow the oil draining, starting production in a well. The PLSVSP objective is to anticipate the completion of the most productive wells. The problem can be seen as a variant of a batch scheduling problem with identical parallel machines and non-anticipatory family setup times to minimize the total weighted completion time. In this analogy, vessels are machines, wells are jobs, and batches are voyages executed by PLSVs, defining which wells to connect each time it leaves the port. We developed several optimization approaches to solve the deterministic and stochastic variants of the problem. For the deterministic problem, we developed hybrid methods and a metaheuristic that outperformed the pure MIP formulations, being practical to deal with the PLSVSP. A simheuristic using embedded Monte Carlo simulation was developed for the stochastic variant of the problem, considering uncertainties in the connection duration and the arrival dates of pipelines at the port. The results show a significant improvement in the solutions dealing with uncertainties compared to solutions generated by a deterministic method. The use of simulation within a metaheuristic framework proved to be a promising approach, being able to deal with the stochastic problem, with little extra computational effort required.

Keywords

Offshore logistics; Ship scheduling; Mathematical formulation; Matheuristic; Simheuristic; Monte Carlo simulation.

Resumo

Cunha, Victor Abu-Marrul Carneiro da; Pinto, Rafael Martinelli; Hamacher, Silvio. **Resolvendo os Problemas Determinístico e Estocástico de Escalonamento de Embarcações do Tipo Pipe-Laying Support Vessel**. Rio de Janeiro, 2021. 154p. Tese de Doutorado – Departamento de Engenharia Industrial, Pontifícia Universidade Católica do Rio de Janeiro.

Empresas de exploração de petróleo e gás *offshore* frequentemente precisam lidar com problemas relacionados ao uso eficiente de seus recursos. Neste trabalho, abordamos um problema de programação de navios associado à logística *offshore* de petróleo e gás – O Problema de Programação de Embarcações do tipo *Pipe-Laying support Vessel* (PLSVSP). Essas embarcações são especialmente projetadas para realizar conexões de dutos entre poços de petróleo submarinos e plataformas de produção. A conexão de dutos é a última etapa a ser executada para permitir a drenagem do óleo e iniciar a produção em um poço. No PLSVSP, o objetivo é antecipar a conclusão de poços mais produtivos. O problema pode ser visto como uma variante de um problema de programação de lotes com máquinas paralelas idênticas e tempos de configuração não antecipados por família para minimizar o *total weighted completion time*. Nessa analogia, embarcações são as máquinas, poços são as tarefas e lotes são as viagens executadas por PLSVs, definindo quais poços devem ser conectados a cada saída do porto. Foram desenvolvidas diversas abordagens de otimização para resolver as variantes determinística e estocástica do problema. Para a variante determinística, desenvolvemos métodos híbridos e uma metaheurística capazes de melhorar as soluções desenvolvidas por formulações MIP puras e lidar com o PLSVSP. Para a variante estocástica, foi desenvolvida uma simheurística utilizando simulação de Monte Carlo incorporada, considerando incertezas nas durações das conexões e nas datas de chegada dos oleodutos no porto. Os resultados mostram uma melhora significativa no custo das soluções quando lidam com incertezas em comparação com soluções geradas por um método determinístico. O uso da simulação em uma estrutura metaheurística mostrou-se uma abordagem promissora, capaz de lidar com o problema estocástico, com pouco esforço computacional extra necessário.

Palavras-chave

Logística offshore; Programação de embarcações; Formulação matemática; Métodos híbridos; Metaheurísticas; Simulação de Monte Carlo.

Table of contents

List of figures

List of tables

List of abbreviations

ALNS – Adaptive Large Neighborhood Search
ATCS – Apparent Tardiness Cost with Setups
BKS – Best Known Solution
CARP – Capacitated Arc Routing Problem
CWS – Clarke & Wright Savings
ERD – Earliest Release Date
E&P – Exploration and Production
FPSO – Floating Production Storage and Offloading
GRASP – Greedy Randomized Adaptive Search Procedure
ILS – Iterated Local Search
IRP – Inventory Routing Problem
LPT – Longest Processing Time
MCS – Monte Carlo Simulation
MCT – Minimum Completion Time
MIP - Mixed Integer Programming
PFSP – Permutation Flow Shop Scheduling Problem
PLSV – Pipe Laying Support Vessel
PLSVSP – Pipe Laying Support Vessel Scheduling Problem
RCL – Restrict Candidate List
RNVD – Randomized Variable Neighborhood Descent
ROV – Remotely Operated underwater Vehicle
RPD – Relative Percentage Deviation
SA – Simulated Annealing
S-PLSVSP – Stochastic Pipe Laying Support Vessel Scheduling Problem
SPT – Shortest Processing Time
TS – Tabu Search
TWCT – Total Weighted Completion Time
VRP – Vehicle Routing Problem
VNS – Variable Neighborhood Search
VND – Variable Neighborhood Descent
WMCT – Weighted Minimum Completion Time
WSPT – Weighted Short Processing Time
2L-VRP – Two-dimensional Vehicle Routing Problem

1
Introduction

The discovery of Brazilian pre-salt fields in 2006 duplicated Brazilian oil and gas reserves. These fields are located in ultra-deep waters, below the ocean salt deposits, exceeding 2,000 meters of water depth. Exploration and Production (E&P) are notably challenging in this region in terms of technology and sustainability (Beltrao et al. 2009, Allahyarzadeh-Bidgoli et al. 2018, Haddad and Giuberti 2010). Wells drilled in the Brazilian pre-salt basin are connected to surface platforms by flexible pipelines that better fit the high water depths. The Pipe-Laying Support Vessel (PLSV) is a ship specially designed to connect pipelines between sub-sea oil wells and production platforms in ultra-deep waters. These vessels are responsible for loading the pipelines at the port, transporting them to the wells' location, laying them out in the ocean, and connecting them between the wells and platforms, allowing production to begin (Speight 2015, Clevelario et al. 2010).

In scheduling optimization problems, decision-makers need to efficiently allocate tasks to usually limited resources, aiming at a specific predetermined objective. In general, these objectives relate to minimizing costs, maximizing productivity, and others (Pinedo 2012). Proper use of resources is even more critical in the offshore oil industry due to the high operating and contract costs of a company's fleet. The *PLSV Scheduling Problem* (PLSVSP) consists of servicing a demand of sub-sea oil wells connections, finding the best schedule for a limited PLSV fleet, anticipating the completion of wells with higher production levels (i.e., wells drilled in larger oil deposits). Each well requires a specific number of pipelines to be connected, being the last stage before the production begins, enabling the oil draining to surface platforms. The daily cost of operating a PLSV is around US$ 300,000, highlighting the importance of efficiently scheduling these resources (SINAVAL 2013, Offshore Energy Today 2013).

In the present work, we study the PLSVSP deterministic and stochastic variants related to a Brazilian oil and gas company that explores the pre-salt basin. In its deterministic version, no uncertainty is taken into account, following the approach that the company currently uses to schedule its contracted PLSV fleet. In the stochastic version, we include uncertainties related to the processing times to perform the pipeline connections and the arrival dates of pipelines at the port. PLSVs are scheduled to execute voyages, where each voyage corresponds to the pipeline loading process at the port, followed by the connections of these pipelines. Navigation times are disregarded because the wells are geographically close to each other in the pre-salt basin. Currently, the PLSVSP is done manually by company specialists, based on their tacit knowledge, without any decision support tool to assist the process. Schedulers must comply with the company's management guidelines and find solutions that match their defined objectives. Our goal is to provide decision support tools for the company to assist the planner in this process. The ability to provide adequate solutions to the real problem is assessed by testing the proposed approaches on a synthetic set of instances generated from real pre-salt data. We use synthetic instances due to confidentiality issues.

The PLSVSP can be seen as a variant of an identical parallel machine scheduling problem with family setup times. In this correspondence, vessels represent the machines, the pipeline connection operations are the set of tasks to be executed by the machines, and the wells are the jobs to complete. A vessel voyage can be interpreted as a batch of tasks to be performed sequentially with a maximum size restricted by the available space on the vessel's deck. The loading times of the pipelines at the port correspond to machine setup times. Setup times are non-anticipatory in the problem since the loading process can only start when all pipelines to connect in a given voyage have arrived in the port. Connections are grouped into families according to similarities in the loading process. Thus, setup times vary depending on the family of each batch. Finally, the machines are called identical as the operations have fixed processing times, regardless of the machine that performs them. The problem has already been addressed as a parallel machine scheduling problem by Queiroz and Mendes (2011) and Cunha et al. (2020). However, the former approach simplifies some of the characteristics, modeling the problem as a classical identical parallel machine scheduling problem. The latter focuses on a rescheduling problem by applying heuristics to minimize impacts caused by disruptions on given schedules considering a set of 10 small instances. None of the works modeled the complete problem. In a similar context in the offshore oil industry, Fernández Pérez et al. (2018) and Monemi et al. (2015) approached the scheduling of rigs, considering realistic aspects of the problem, but not dealing with family-based setup times.

In this work, we use optimization techniques to solve the PLSVSP. The thesis has two main optimization parts. The first concerning the deterministic PLSVSP introduces four mathematical formulations, several constructive heuristics using dispatching rules based on the machine scheduling literature, two matheuristics using mathematical formulations with MIP-based neighborhood searches, and one metaheuristic. This part aims to provide and compare different optimization techniques to solve the problem providing a good perspective about vantages and disadvantages in considering each approach. Also, this part presents a benchmark with 72 instances generated from real pre-salt data. The idea is to provide a set of instances for the scheduling community so that interested researchers could work on the problem, extending or proposing new algorithms to solve it. The second optimization part within the thesis introduces a simulation-optimization method with embedded Monte Carlo simulation, called simheuristic, to deal with the stochastic PLSVSP with uncertainties regarding the operation's processing times and pipeline arrival dates. This part aims to propose a method that better fits the realistic process with its uncertainties, providing a more reliable statistic evaluation of the solutions.

1.1 Contributions

The thesis's objective is to show the advantages of using optimization techniques to solve a complex real-life scheduling problem. Since we are dealing with a complex scheduling problem, it is worth emphasizing that other researchers can apply the developed approaches to simplified variants of the problem. This aspect enhances its relevance to the scheduling literature, supporting studies on similar problems or simplified variants. Our main contributions in this work are threefold: (1) Define the PLSVSP properly, drawing its relationship with the machine scheduling literature; (2) Introduce mathematical formulations to represent the complete problem; (3) Develop optimization algorithms to solve the problem in its deterministic and stochastic variants. The specific contributions of this thesis are summarized below:

- Extend a problem related to an identical parallel machine scheduling problem with family setup times, including realistic aspects, making it more challenging to solve.
- Develop a PLSVSP benchmark instance set based on the studied company's real data regarding the pre-salt exploration layer.
- Address a realistic scheduling problem concerning critical offshore resources by formally defining the problem for the scheduling community.
- Provide four new Mixed Integer Programming (MIP) formulations for the PLSVSP based on the machine scheduling literature.

- Develop several constructive heuristics to generate solutions for PLSVSP with low computational time.
- Develop two new MIP-based neighborhood searches with two mathematical formulations, testing its efficiency within metaheuristic frameworks.
- Increase the literature on matheuristics applied to scheduling problems.
- Introduce an Iterated Greedy algorithm to solve the PLSVSP, extending an Iterated Local Search metaheuristic from the literature.
- Address a stochastic version of the PLSVSP considering uncertainties in the pipeline connection processing times and their arrival dates at the port.
- Develop a simheuristic using embedded Monte Carlo simulation to deal with the stochastic PLSVSP.

1.2 Thesis Structure

This thesis is organized into 8 chapters, including this introductory chapter. Chapter 2 provides a review of the literature on the main subjects covered in the thesis. It includes an overview of the PLSV logistics and ship scheduling problems and a review of papers dealing with identical parallel machines scheduling problems, matheuristics to solve scheduling problems, and simheuristics applied to several logistic problems. Chapter 3 presents a complete description of the PLSVSP, including its stochastic version and its relationship with an identical parallel machine scheduling problem. From Chapter 4 to 8 we introduce the optimization approaches developed to deal with the PLSVSP. These chapters include the description of the approach, experimentation, and results analysis. As the main objective of the thesis is to develop and test different optimization techniques to solve the PLSVSP, we present in Figure 1.1 a diagram that connects the methods to help the reader to follow the structure of the thesis.

The first two parts, at the top of the diagram, are the mathematical formulations (Chapter 4) and the constructive heuristics (Chapter 5). Chapter 4 presents three mathematical formulations for the PLSVSP based on the machine scheduling literature and provides a benchmark of 72 PLSVSP instances generated from real data from the Brazilian pre-salt basin. In chapter 5, several constructive heuristics are introduced, combining machine scheduling dispatching rules and task weight estimation rules with machine assignment and batch composition steps. Chapter 6 presents a new formulation for the

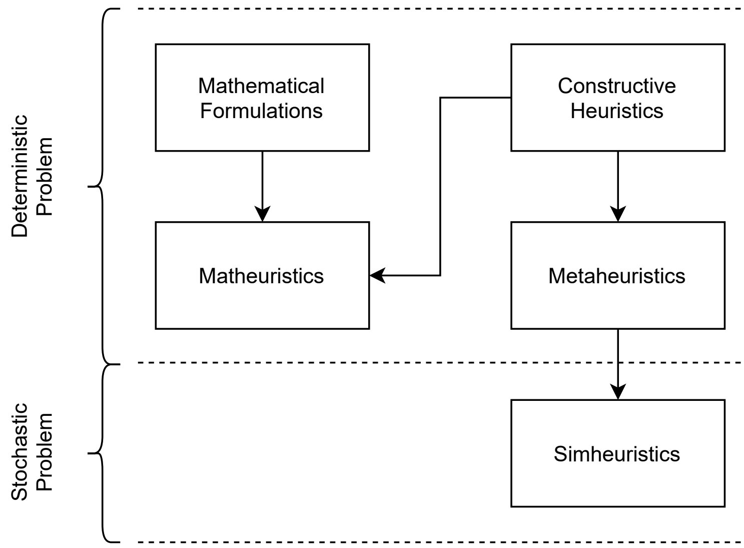


Figure 1.1: General optimization methods diagram.

PLSVSP and two matheuristic approaches using two new MIP-based neighborhood searches combined with two metaheuristic frameworks. A constructive heuristic is used to initialize the matheuristics and the metaheuristic introduced in Chapter 7. The metaheuristic is described in detail in Chapter 7, including the neighborhood structures used in the local search step. Finally, Chapter 8 considers the stochastic PLSVSP. In this chapter, the metaheuristic is extended to a simheuristic with the inclusion of simulation steps to identify promising stochastic solutions.

Chapter 9 concludes the thesis and presents some perspectives for future works regarding the PLSVSP and some of the optimization approaches presented.

2
Literature Review

In this chapter, we present a review of the different subjects involved in the thesis. The chapter is divided into five parts. In the first part, we provide an overview of Brazilian offshore logistics to help understanding the PLSV importance in the process. An overview of ship scheduling problems is provided in the second part, highlighting the differences between this class of problems and the PLSVP. In the third part, a review of parallel machine scheduling problems is presented, focusing on problems with family setup times. In the fourth part, we review heuristics applied to scheduling problems. The last part of the chapter gives a review of simheuristics applied to combinatorial optimization problems.

2.1
Brazilian Offshore Logistics

To contextualize the problem and provide a better view of the role of PLSV in the exploration and production of oil and gas, we describe, in this section, some properties of Brazilian offshore logistics.

The life cycle of oil operations includes exploration and development, production, refining, marketing, transportation, and final utilization. Such activities divide the industry into two major segments: upstream, related to exploration and production activities, and downstream, related to the refining and marketing of oil and its derivatives. In the exploration phase, geological and geophysical data are analyzed to identify potential oil production sites. This phase is crucial to reduce the costs associated with well-drilling tasks (Thomas 2001, Islam and Khan 2013, Devold 2013).

The appraisal, drilling, and completion of offshore sub-sea wells are carried out by maritime rigs, responsible for preparing them to operate. In the last stage, the rig installs the sub-sea tree, equipment employed for regulating the flow of a well through an assembly of valves, spools, and fittings. The oil flows through pipelines connected between the sub-sea tree and the production platform. The PLSVs operate after the sub-sea tree's installation, connecting the pipelines, allowing the well to start producing. PLSVs are also responsible for transporting the pipelines to be launched into the sea, and connecting then by using a Remotely Operated underwater Vehicle (ROV), equipped inside the vessel (De Lima 2007, Thomas 2001).

Several oil discoveries in the Brazilian basin have been released in recent years. Most of them in ultra-deep waters below the salt layer called the pre-salt layer. These discoveries duplicated Brazilian oil and gas reserves, increasing the country's relevance in the global oil industry. In this exploratory region, flexible oil pipelines are used to connect underwater oil wells. The structure of these pipelines is variable, which is defined based on detailed engineering projects. The pipelines are designed to withstand the conditions of pressure and depth that will be subjected. In addition to the possibility of their use in ultra-deep waters, the flexible pipelines are easily transported within the PLSVs. (LABANCA 2005, Rodrigues and Sauer 2015).

2.2 Ship Scheduling

The world's maritime fleet, which is close to 90.000 ships, is responsible for the transport of around 80% of international commerce, emerging as the most crucial modal for global trade (UNCTAD 2015). Purchasing a vessel requires a high investment, in addition to high operating costs. These aspects highlight the importance of scheduling these resources efficiently, optimizing their utilization (Christiansen et al. 2007).

Ship Scheduling problems are classified into three modes, not mutually exclusives (Lawrence 1972): liner, tramp, and industrial. A liner shipping company operates with fixed routes for vessels. Tramp companies work by contracts and optional cargo transportation. In industrial shipping, the company controls the ships. These operators aim to attend their demands with minimal cost. Relevant reviews were published on the subject at intervals of approximately ten years between them. For interested readers, we recommend Ronen (1983), Ronen (1993), Christiansen et al. (2004), and Christiansen et al. (2013) for a good overview of ship scheduling problems. These problems are a class of vehicle routing problems with some additional characteristics regarding the maritime environment.

Although PLSVSP does not regards to cargo transportation, Mendes (2007) points out that, as the company controls its fleet, it can be classified as an industrial operator. It is worth mentioning that many similarities can be found between routing and scheduling problems, as highlight by Beck et al. (2002), Beck et al. (2003), and Kouki et al. (2007).

This section helps to understand how PLSVSP differs from ship scheduling problems. In the PLSVSP, we follow the studied company's approach that disregards travel times due to the proximity of oil wells in the pre-salt basin and the discrepancy between travel times and the time spent to perform pipeline connections. In general, travel times take a few hours, while connection operations last for days. Thus, we look for similar problems in the literature, where the correspondence with scheduling problems related to critical resources, generalized as machines, was identified. The next section presents a review of parallel machine scheduling problems with characteristics similar to the PLSVSP.

2.3 Parallel Machine Scheduling Problem

The PLSVSP can be seen as a variant of a parallel machine scheduling problem with family setup times. This class of problems involves meeting a given demand for tasks to be performed by a given set of machines in parallel, optimizing these resources. In these problems, planners must make two main decisions: the assignment of machines to perform the tasks and the sequence in which the machines must perform the tasks. When the machines are identical, as, in the case of the PLSVSP, each task has a fixed processing time, i.e., they are independent of the machine assigned to perform the task (Su 2009, Gokhale and Mathirajan 2012). Furthermore, when family setup times are considered, it means that a setup time dependent on the family of the tasks must be included in the schedule whenever the machine changes the execution of tasks from different families (Allahverdi 2015). The sequence of tasks of the same family sharing the same setup time is called a batch. Setup times are non-anticipatory when their start depends on the release dates of the tasks in the batch. Below, we revise papers that deal with parallel machine scheduling problems with family setup times, the same class of problems as the PLSVSP. At the end of the section, we include some documents from other machine environments that deal with non-anticipatory setup times. From the best of our knowledge, no work addresses problems with non-anticipatory family setup times.

Webster and Azizoglu (2001) proposed backward and forward dynamic program algorithms to minimize the total weighted completion time. They show that when the number of machines and families are fixed, the former algorithm is polynomial in the sum of the weights and the latter in the sum of processing and setup times. Azizoglu and Webster (2003) tested four different branch-and-bound algorithms with the same objective with two different approaches to solve the problem. The first one decomposes the problem into two phases, initially generating a solution without setups and adding them later. The second approach considers the setup times as jobs to be scheduled.

The approach is based on creating an ordered list of jobs to be scheduled, using the concept that an optimal schedule can be obtained from one specific order on this list, by assigning each job in the list in the given sequence to the machine that finishes it first. Dunstall and Wirth (2005a) proposed a new branch-and-bound scheme derived from an application for a problem without family setup times. Bettayeb et al. (2008) applied the same branching scheme improving the generation of lower bounds from Dunstall and Wirth (2005a) approach. They have reduced the number of visited nodes, concluding to be a good approach for large-sized instances. Omar and Teo (2006) focused on minimizing the total weighted earliness and tardiness, where the concept of this objective function is to completes the execution of all jobs closer to their due dates, known as a just in time objective. They developed a new mathematical formulation for the problem, capable of solving instances with up to 18 jobs, four machines, and four families. Chen and Powell (2003) studied two problems, the first one with the sequence-independent family setup times to minimize the total completion time and the second one considering sequence-dependent setup times to minimize the total number of tardy jobs. A branch-and-bound, in conjunction with column generation, was used to find solutions to problems with up to 40 jobs, four machines, and six families.

As in the case of the PLSVSP, other real-life problems are also included in this class of machine scheduling problems. Shin and Leon (2004) addressed a real scheduling problem related to the semiconductor industry, focusing on minimizing the total tardiness of the jobs. They applied a two-phase heuristic procedure, generating solutions using a bin-packing approach and improving them by applying a tabu search. Schaller (2014) improved the method proposed by Shin and Leon (2004) with three new tabu searches and two genetic algorithms. Obeid et al. (2014) also studied a scheduling problem in the semiconductor industry proposing two mathematical formulations and two heuristics considering that the eligibility of a machine to process a job might change over time according to the schedule. Ciavotta et al. (2016) applied a more general framework with a roll-out algorithm using several dispatching rules simultaneously to solve a real-life problem, considering release dates, due dates, hard deadline constraints for jobs, maximum campaign size and unavailable periods. They tested several objective functions solved in a given lexicographic order. These papers with a realistic background are the ones that deal with the most complex scheduling problems. However, none of them considered the specific characteristics approached in the PLSVSP.

Additionally, many works applied heuristics and metaheuristics to this class of problem. Dunstall and Wirth (2005b) proposed several heuristics evaluating their performance with lower bounds and optimal solutions to minimize the total weighted completion time. Liao et al. (2012) continued Dunstall and Wirth (2005b) work by improving their heuristics, proposing some benchmark instances. From this benchmark, Tseng and Lee (2017) tested a metaheuristic based on electromagnetism concepts, also comparing the results with a genetic algorithm. Mehdizadeh et al. (2015) proposed a vibration dumping optimization algorithm comparing their solutions with a genetic algorithm and a MIP formulation introduced by Tavakkoli-Moghaddam et al. (2007) for the same problem. Eom et al. (2002) developed an efficient heuristic to minimize the total weighted tardiness, based on the Apparent Tardiness Cost with Setups (ATCS) composite rule (Lee and Pinedo 1997), with sequence-dependent family setup times. Van Der Zee (2015) developed a survey on dispatching methods to address parallel machines scheduling problems with family setup times.

Another challenging characteristic of the PLSVSP regards the non-anticipatory setup consideration. Less studied in the scheduling literature, this aspect has been applied more often in other machine shop environments. Fonseca-Reyna et al. (2019), Ruiz et al. (2008), Lin and Cheng (2005) and Fuchigami et al. (2015) considered this aspect within different flow shop problems, while González et al. (2015) and Roshanaei et al. (2010) approached this aspect within a job-shop scheduling environment. Lin and Cheng (2005) was the only one among those to consider the setup time while scheduling batches. However, they consider a constant setup time for each batch and not family-based ones.

Valuable reviews focusing on parallel machine scheduling problems, with different aspects addressed, are found in the literature. We refer the interested reader, to the works of Cheng and Sin (1990), Lam and Xing (1997), Mokotoff (2001), Li and Yang (2009) Behera (2012), Edis et al. (2013), and Kaabi and Harrath (2014). For a comprehensive overview of problems with setup considerations or batching, the reader is referred to Allahverdi (2015) and Potts and Kovalyov (2000).

2.4 Matheuristics

Matheuristics are hybrid approaches that combine concepts of meta-heuristics and exact methods to solve combinatorial optimization problems, being a growing field in the literature due to the improvement of computers and solvers (Thompson 2018). Some researchers applied matheuristics to solve machine scheduling problems.

Billaut et al. (2015) handled a single machine environment using a two-step approach, combining a beam search algorithm with a MIP-based neighborhood search. Regarding parallel machines environments, Ekici et al. (2019) applied a Tabu Search matheuristic, prohibiting some job-machine assignments during its execution. Ozer and Sarac (2019) developed a two-step approach combining a genetic algorithm with a MIP model. Woo and Kim (2018) proposed a two-step approach, grouping jobs in so-called buckets using metaheuristics and assigning them to machines by a mathematical model. Fanjul-Peyro et al. (2017) introduced matheuristics using constraint relaxation, limiting job-machine assignments, and using a MIP-based neighborhood search to optimize subsets of jobs. Other researchers applied matheuristics to solve flow-shop scheduling problems. Ta et al. (2018), Della Croce et al. (2014), and Della Croce et al. (2019) used MIP-based neighborhoods in positional scheduling formulations, solving sub-problems for a limited number of positions. Lin and Ying (2016) and Lin and Ying (2019) converted the flow-shop problem into a traveling salesman problem, building an initial heuristic solution, and solving a mathematical model. To our knowledge, Mönch and Roob (2018) is the only work that applies matheuristics to a batch scheduling formulation. However, they do not apply MIP-based neighborhood search, but use a two-step approach that combines a genetic algorithm to compose batches and a mathematical model to assign and sequence them on the machines.

Matheuristics have also been successfully applied to many scheduling problems with realistic backgrounds. See, for instance, the works of Martinelli et al. (2019), Kalinowski et al. (2020), and Grenouilleau et al. (2020), related to the scheduling of mining activities, rail network maintenance, and home health care services, respectively.

2.5
Simheuristics

Simheuristics are simulation-optimization approaches that combine metaheuristic frameworks with embedded simulation to solve stochastic combinatorial optimization problems in reasonable computational times. These methods take advantage of the structure and elements of regular metaheuristics, based on the premise of a correlation between good deterministic and stochastic solutions, to identify and evaluate promising solutions through a simulation step. Simheuristic approaches can provide more statistical information about a solution, which helps for risk-analysis purposes. It is a growing field in the literature with successful applications for several problems, from vehicle routing problems (VRP) to scheduling problems (Juan et al. 2015).

Juan et al. (2014) addressed a Permutation Flow Shop Scheduling Problem (PFSP) with stochastic processing times. The authors developed a simheuristic, named SimILS, using Monte Carlo Simulation (MCS) within an Iterated Local Search (ILS) approach, testing three uncertainty levels. They showed the advantages of using a simheuristic approach compared with the deterministic solutions in the stochastic environment. Gonzalez-Neira et al. (2017) dealt with the same problem but using a simheuristic within a Greedy Randomized Adaptive Search framework. The authors compared the deterministic solution with two stochastic solutions, one with the best average objective function value and another with the smallest standard deviation. Hatami et al. (2018) focused on a parallel flow shop problem with deadlines and stochastic processing times. They applied two SimILS algorithms to minimize the expected makespan and the makespan percentile. In the experiments, they analyze the probability of accomplishing the deadlines of a given solution. González-Neira et al. (2019) developed a Tabu Search simheuristic to solve a stochastic PFSP. They tested two different probability distributions (Log-normal and uniform) for the processing times with three levels of uncertainty, analyzing solutions according to the expected tardiness and standard deviation. Villarinho et al. (2021) also addressed a PFSP with stochastic processing times but aiming to maximize the schedules' expected payoffs. The authors developed a multi-start simheuristic using risk analysis in their experiments to quantify the worst-case scenarios.

Quintero-Araujo et al. (2017) tackled a VRP under demand uncertainty, comparing a collaborative approach with a non-collaborative one. They generated a set of promising solutions and evaluated the reliability of the routes to accomplish the demand. Gruler et al. (2017) also dealt with a VRP with uncertainty on demands for a waste collection problem. They applied a Variable Neighborhood Search (VNS) with MCS, considering different safety capacity levels on the vehicles, evaluating the reliability and routing costs of a pool with the best stochastic solutions. Guimarans et al. (2018) approached a two-dimensional VRP using a SimILS with biased randomization and a simulated annealing acceptance criterion, also returning a pool with the best stochastic solutions. Reyes-Rubiano et al. (2019) applied a simheuristic using biased randomization to tackle a VRP with electric vehicles and uncertainty on travel times. They defined different safety stock levels to evaluate the total cost, considering routes and vehicles' reliability to complete routes without energy failure. Calvet et al. (2019) dealt with a multi depot VRP with stochastic demands and limited capacity. They compared a two-stage stochastic-programming approach with different simheuristics showing the advantages of using a SimILS method. Gonzalez-Martin et al. (2018) solved a Capacitated Arc Routing Problem with demand uncertainty, evaluating the routes' reliability by joining an MCS with a biased randomized method. Yazdani et al. (2021) developed a simheuristic based on a hybrid Genetic Algorithm to solve a real-life waste collection VRP. Latorre-Biel et al. (2021) considered correlated stochastic demands within a capacitated VRP, combining a simheuristic with a Petri net predictor to update the mean demands.

Gruler et al. (2018) studied an Inventory Routing Problem (IRP) with stochastic demands. They applied a simheuristic in a VNS approach, using biased randomization with a Clarke & Wright Savings heuristic to build initial solutions. They evaluate several refilling policies evaluating holding and stockout costs. Onggo et al. (2019) developed a simheuristic to tackle an IRP with perishable products to minimize the trade-off between the total operational cost and the food-waste cost. Raba et al. (2020) also approached an IRP with stochastic demands using a reactive strategy to reevaluate the refilling policies periodically. Gruler et al. (2020) developed a VNS to solve an IRP with stochastic demands. They run the VNS to build a pool with the best deterministic solutions, using an MCS to evaluate these solutions in the stochastic environment further. Quintero-Araujo et al. (2019) applied a SimILS to deal with a capacitated Location-Routing Problem (LRP) with stochastic demands. They evaluate reactive and preventive strategies while defining routes to minimize the impacts of route failure. Rabbani et al. (2019) developed a genetic algorithm with MCS to optimize a stochastic hazardous waste LRP aiming to minimize the total operational cost and the contamination risks according to the defined locations and routes. Pagès-Bernaus et al. (2019) compared a SimILS with a two-stage stochastic-programming approach to solve a Facility Location Problem with uncertain demands.

Lopes et al. (2020) developed a SimILS with a specialized cycle time simulator to address a balancing optimization problem for an assembly line, with stochastic sequences of products, evaluating several buffer layouts. Santos et al. (2020) also designed a simheuristic based on the ILS approach to increase the production rate of a Brazilian mining company. They combined the metaheuristic framework with an operational simulator, responsible for evaluating the solutions' objective function values, for defining the amount of active equipment within the company plant. Fabri and Ramalhinho (2021) also developed a SimILS to define the supplying routes from the warehouse to the workstations in a car assembling production line. The authors used the simulation step to identify the number of backorders for each simulated period and compute the objective function of the solutions.

Panadero et al. (2020) designed a VNS simheuristic with a simulated annealing acceptance criterion to deal with a project portfolio selection problem with maximum risk constraint and uncertainties in cash flows and discount rates. They developed a set of instances with different interdependencies between the projects, modeling the uncertainties with normal distributions, aiming to maximize the portfolio's expected net present value.

Note that VRP variants are among the most addressed problems in the simheuristic literature, followed by Flow Shop problems. It is worth mentioning the relevance in applying simheuristics as it is a growing field in the literature, with most of the works published in the last five years. Concerning works that deal with scheduling problems, usually, the authors consider the uncertainty only in the duration of the tasks (processing times), using, in general, the same probability distribution with a specific variance level for all tasks. Also, to the best of our knowledge, no article addresses ship scheduling problems or parallel machine scheduling problems.

3
Problem Description

This chapter is organized into three sections. First, we detail the real problem of scheduling the PLSV fleet, providing more information about the process. The second section provides a more formal description of the problem, using the correlation with a parallel machine scheduling problem. This section includes an illustrative example of a PLSV schedule to help the reader better understand the problem. Finally, the stochastic version of PLSVSP is presented.

3.1
PLSV Scheduling Problem

The PLSVSP consists of scheduling a given PLSV fleet to meet a demand for pipeline connections in different subsea oil wells. The objective is to anticipate the completion of more productive wells. Our approach is related to a Brazilian oil and gas company that operates in the pre-salt exploratory basin. As mentioned before, PLSVs are responsible for loading these pipelines at the port, transporting them to the wells site, laying them out in the ocean, and connecting them between the wells and platforms, allowing production to begin. Figure 3.1 shows an PLSV. For a better understanding of the problem, we first provide more details about the main PLSVSP actors in the following:

Wells: Based on management guidelines, the company defines the next wells that must be completed, identifying the pipelines to connect in each specific well. A well is only ready to produce when all of its pipelines are connected. Also, the company has an estimated production rate for each well.

Pipeline Connections: Each connection concerns a specific pipeline and well. The company estimates the duration of the connections based on historical data, taking into account several aspects such as distance between the well and the platform, water depth, and others. Another relevant information regards the space that each pipeline occupies on the deck of a PLSV vessel. Figures 3.2 shows the flexible pipelines used in the Brazilian pre-salt region, and Figure 3.3 depicts a scheme with the pipelines connected between the wells and the platform.

PLSV fleet: To perform the connections, the company has a heterogeneous fleet of PLSVs. Each vessel is available to operate from a specific date. Besides, each vessel has a different capacity and is eligible to serve only a subset of connections due to the heterogeneity of the fleet.

Figure 3.1: Pipe Laying Support Vessel (PLSV) O Petróleo (2017).

Figure 3.2: Flexible Pipelines National Oilwell Varco (2020).

Due to basin characteristics, each connection affects the pressure in the oil & gas reservoir, impacting the production of other wells in the same region. Thus, the company assumes that a well only begins to produce with the expected potential when, in addition to its connections, all those that impact its production are also completed. Consequently, some operations are associated with several wells simultaneously, creating intersections on the subsets of connections associated with the wells.

When executing a connection, a PLSV goes to the port to load the pipeline on its deck, travels to the well's location, launches the pipelines into the ocean, and connects them between the well and the platform. Navigation times are disregarded due to the proximity between the wells in the pre-salt basin.

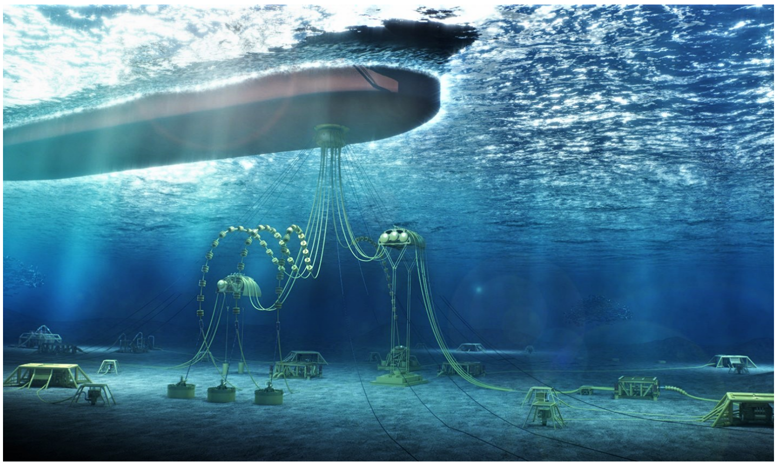


Figure 3.3: Subsea production system scheme, showing a Floating Production Storage and Offloading (FPSO) platform connected to several oil wells by flexible pipelines. TecPetro (2015).

Thus, the duration of a pipeline connection takes into account the time spent laying the pipeline at the ocean floor and connecting it. The combination of the loading process at the port with the connection task defines a PLSV voyage. In one voyage, a PLSV can connect several pipelines from different wells. However, according to the occupation of the pipelines, the vessel's capacity limits the number of connections to execute. Moreover, the loading process at the port can only begin when all pipelines to connect on a given voyage are available at the port, that is, anticipations for loading some pipelines for a given voyage are not allowed.

As previously mentioned, the scheduling process is performed by specialists from the company based on their experience. The company groups connections according to the similarity in their pipeline loading process at the port. The time spent loading the pipelines is dependent on the group of connections to execute on a voyage, with a fixed duration, i.e., they do not depend on the number of connections to be performed. Three main decisions are made by the specialists when scheduling a PLSV fleet:

1. Definition of the voyages

2. Assignment of the voyages to the vessels

3. Sequencing of the voyages on each vessel

In the first step, the planners must only build voyages with connections of the same group. On the assignment step, the schedulers must check if a vessel is eligible and has enough space on the deck to perform all pipeline connections defined for a voyage. Finally, on the voyage sequencing step, the planners must consider the arrival of the pipelines at the port, which may occur on different days, and the day in which each vessel is available for starting their activities. Considering the non-anticipation rule, the defined sequence directly affects the idleness of the vessels and the delay in starting subsequent voyages. Due to the problem constraints and the concern with the solution's quality, all activities from this decision process are carried out simultaneously.

The PLSVSP is described in the next section as an identical parallel machine scheduling problem with non-anticipatory family setup times. In this machine environment, a set of parallel machines is available to perform a given set of tasks. Machines are called identical when tasks' processing times are fixed and, therefore, machine-independent (Pinedo 2012). In most studies, tasks are called jobs, as each task requires only a single operation to complete. However, we represent tasks as operations, given that, in our problem, a set of operations must be finished to complete a job. Besides, when family setup times are considered, tasks are grouped into families by similarity, and a setup time must be scheduled whenever a machine changes the execution of a task from one family to another. In this class of problems, the combination of one setup time and its subsequent tasks is called a batch. Also, when these setup times are non-anticipatory, the starting time of a batch cannot be anticipated. That is, it depends on the release of the tasks assigned to it.

In the PLSVSP, vessels are machines, jobs represent wells, and pipeline connections are the operations. The non-anticipatory family setup times represent pipeline loading times, while PLSV voyages are batches. Next section formalizes the problem as a parallel machine scheduling problem and provides a mapping between the machine scheduling aspects and the PLSVSP context.

3.2 Identical Parallel Machine Scheduling Approach

The notation and assumptions considered in the PLSVSP are described below, according to the machine scheduling theory.

1. There is a set $\mathcal{O} = \{O_i | i = 1, ..., o\}$ of operations, where each operation O_i has a processing time p_i, a release date r_i, a load occupancy or size l_i, and a family f_i.

2. There is a set $\mathcal{F} = \{F_g \mid g = 1, ..., f\}$ of families where s_g is the setup time for a family F_g.

3. All operations must be scheduled without preemption in a set $\mathcal{M} = \{M_k \mid k = 1, ..., m\}$ of identical machines.

4. Each machine $M_k \in \mathcal{M}$ is available to process operations from its release date r_k and has a capacity q_k.

5. Machines are called identical due to the fixed processing times of the operations.

6. A subset $\mathcal{M}_i \subseteq \mathcal{M}$ defines the eligible machines for executing each operation O_i. Conversely, $\mathcal{O}_k$ is a subset of operations $O_i \in \mathcal{O}$ that a machine $M_k \in \mathcal{M}$ is eligible to execute, i.e., $\mathcal{O}_k = \{O_i \in \mathcal{O} \mid M_k \in \mathcal{M}_i\}$.

7. A family setup time is incurred on three occasions: while changing the execution of operations from different families, before the first operation on each machine, or when the machine's capacity is reached.

8. We define *Batch* as a combination of one family setup time followed by a sequence with one or more operations from the same family.

9. The batching mode is Serial-Batching. Thus, the processing time of a batch is given by the sum of the operations' processing times within the batch plus the setup time duration regarding the batch family.

10. The size of a batch, computed by the sum of the load occupancy of the operations within it, must respect the capacity of the machine assigned to execute it.

11. The setup times are non-anticipatory, i.e., a *Batch* can only start when all operations within it are released.

12. There is a set $\mathcal{N} = \{J_j \mid j = 1, ..., n\}$ of jobs where each job J_j is associated with a subset $\mathcal{O}_j \subseteq \mathcal{O}$ of operations and has a weight w_j defining its priority.

13. We use $\mathcal{N}_i \subseteq \mathcal{N}$ to identify a subset of jobs associated with operation O_i, since in the PLSVSP, one operation might be related to several jobs simultaneously.

14. A job is completed when all of its associated operations are concluded. Thus, the completion time (C_j) of a job is the maximum completion time of the associated operations ($C_j = \max\limits_{O_i \in \mathcal{O}_j} \{C_i\}$). C_i is the completion time of operation O_i.

15. The objective function is to minimize the total weighted completion time of all jobs, defined as $\sum\limits_{J_j \in \mathcal{N}} w_j C_j$.

Table 3.1 provides a mapping between the machine scheduling aspects and the PLSVSP context for the sake of convenience.

Table 3.1: The mapping between the parallel machine scheduling problem definitions and its correspondences with the PLSVSP.

Name	Machine scheduling definition	PLSVSP correspondence
$\mathcal{O}$	Operations	Pipeline connections
$\mathcal{N}$	Jobs	Wells
$\mathcal{M}$	Machines	PLSVs
$\mathcal{F}$	Families	Groups of pipeline connections with similar loading process at the port
$\mathcal{M}_i$	Machine eligibility subset	Subset of vessels eligible to execute a pipeline connection
$\mathcal{O}_k$	Subset of operations a machine is eligible to execute	Subset of connections a vessel can carry
$\mathcal{O}_j$	Subset of operations composing a job	Subset of connections required to enable a well to start producing
$\mathcal{N}_i$	Subset of jobs associated to an operation	Subset of wells that depends on a connection to be able to produce
p_i	Processing time of an operation	Time taken to perform a pipeline connection
r_i	Release date of an operation	Arrival date of a pipeline at the port
l_i	Load occupancy or size of an operation	Pipeline occupancy on the deck of the ship
f_i	Family of an operation	Group of a pipeline connection
r_k	Release date of a machine	Availability date of a vessel
q_k	Capacity of a machine	Vessel's deck capacity
w_j	Weight of a job	Production potential of a well
C_i	Completion time of an operation	Time when a connection ends
C_j	Completion time of a job	Time when a well is fully connected and able to start producing
s_g	Setup times of a family	Time spent at port loading pipelines for a specific group of operations
Batch	Sequence of operations from the same family sharing the same setup time	PLSV voyage

To clarify the batch concept and the non-anticipatory setup times, we depict in Figure 3.4 a schedule example with one batch, containing a generic setup time (s) with 10 days of duration followed by two operations (O_1 and O_2) with equal processing times ($p_1 = p_2 = 20$), assigned to one machine (M_1). Thus, the batch processing time is 50 days. The machine is available from $t = 5$ and processes the batch from $t = 10$ to $t = 60$. The five days idleness is generated by the non-anticipatory setup consideration, which forces the starting time of the setup time (marking the beginning of the batch) to accomplish the operations' release dates (the release dates in the example are $r_1 = 0$ and $r_2 = 10$).

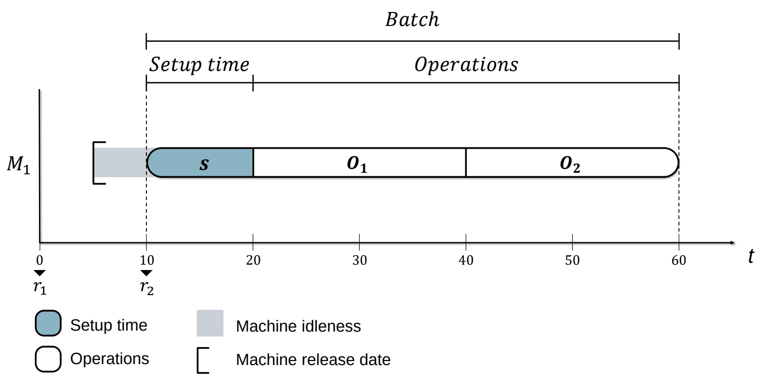


Figure 3.4: PLSV Scheduling example with one voyage assigned to a single machine.

3.2.1 Example Problem

An example is provided below to help illustrating the PLSVSP. We consider an instance with 15 operations, 5 jobs and 4 machines, which means that $o = 15$, $n = 5$ and $m = 4$. Table 3.2 shows the characteristics of each operation O_i in the given instance in the following order: processing time (p_i), release date (r_i), family (f_i), load occupation or size (l_i), set of associated jobs ($\mathcal{N}_i$) and eligible set of machines ($\mathcal{M}_i$). Table 3.3 indicates the weight (w_j) and the sets of operations ($\mathcal{O}_j$) that composes each job J_j. Table 3.4 presents the characteristics of each machine M_k in the given instance in the following order: release date (r_k) and capacity (q_k), eligible operations ($\mathcal{O}_k$). Finally, Table 3.5 shows the characteristics of each family F_g in the given instance in the following order: setup duration (s_g) and sets of operations ($\mathcal{O}_g$) that belongs to family F_g.

Table 3.2: Operations characteristics of the example instance.

Operation (O_i)	p_i	r_i	f_i	l_i	$\mathcal{N}_i$	$\mathcal{M}_i$
O_1	23	5	1	90	$\{J_5\}$	$\{M_1, M_4\}$
O_2	5	17	1	90	$\{J_5\}$	$\{M_1, M_3, M_4\}$
O_3	25	16	3	70	$\{J_2, J_3, J_4\}$	$\{M_2, M_3, M_4\}$
O_4	10	9	2	50	$\{J_1\}$	$\{M_1, M_2, M_3, M_4\}$
O_5	4	18	1	60	$\{J_3\}$	$\{M_1, M_2, M_3, M_4\}$
O_6	8	17	2	80	$\{J_1, J_3\}$	$\{M_2, M_4\}$
O_7	29	15	3	60	$\{J_5\}$	$\{M_1, M_2, M_3, M_4\}$
O_8	8	5	3	0	$\{J_2\}$	$\{M_2\}$
O_9	17	9	3	40	$\{J_1, J_5\}$	$\{M_1, M_2, M_4\}$
O_{10}	25	0	1	40	$\{J_2, J_5\}$	$\{M_1, M_2, M_3, M_4\}$
O_{11}	4	3	1	30	$\{J_3\}$	$\{M_1, M_2, M_3, M_4\}$
O_{12}	15	0	1	20	$\{J_1, J_3\}$	$\{M_1, M_3, M_4\}$
O_{13}	7	16	2	60	$\{J_3\}$	$\{M_3, M_4\}$
O_{14}	7	7	1	90	$\{J_4, J_5\}$	$\{M_1, M_4\}$
O_{15}	18	15	3	20	$\{J_3\}$	$\{M_4\}$

Table 3.3: Jobs characteristics of the example instance.

Job (J_j)	w_j	$\mathcal{O}_j$
J_1	46	$\{O_4, O_6, O_9, O_{12}\}$
J_2	40	$\{O_3, O_8, O_{10}\}$
J_3	39	$\{O_3, O_5, O_6, O_{11}, O_{12}, O_{13}, O_{15}\}$
J_4	13	$\{O_3, O_{14}\}$
J_5	3	$\{O_1, O_2, O_7, O_9, O_{10}, O_{14}\}$

Table 3.4: Machines aspects regarding the example instance.

Machine (M_k)	r_k	q_k	$\mathcal{O}_k$
M_1	13	90	$\{O_1, O_2, O_4, O_5, O_7, O_9, O_{10}, O_{11}, O_{12}, O_{14}\}$
M_2	0	80	$\{O_3, O_4, O_5, O_6, O_7, O_8, O_9, O_{10}, O_{11}\}$
M_3	0	90	$\{O_2, O_3, O_4, O_5, O_7, O_{10}, O_{11}, O_{12}, O_{13}\}$
M_4	1	90	$\{O_1, O_2, O_3, O_4, O_5, O_6, O_7, O_9, O_{10}, O_{11}, O_{12}, O_{13}, O_{14}, O_{15}\}$

Table 3.5: Families characteristics of the example instance.

Family (F_g)	s_g	$\mathcal{O}_g$
F_1	5	$\{O_1, O_2, O_5, O_{10}, O_{11}, O_{12}, O_{14}\}$
F_2	7	$\{O_4, O_6, O_{13}\}$
F_3	9	$\{O_3, O_7, O_8, O_9, O_{15}\}$

In Figure 3.5, the optimal solution of the example instance is presented in a Gantt chart with the respective operations assigned, sequenced, and forming batches. Jobs associated with each operation are described below each allocation. We also marked the respective completion times for each one of the jobs, and setup times are indicated with the family.

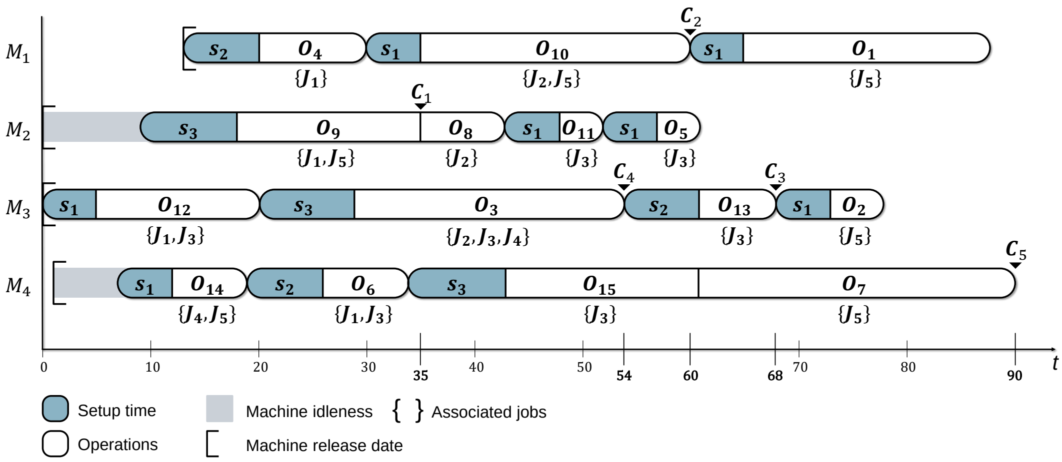


Figure 3.5: Scheduling example with 15 operations, 5 Jobs and 4 machines.

Note that only two batches with more than one operation were formed, both from family F_3. The first on machine M_2, consisting of a setup s_3, and operations O_9 and O_8 with a total load of 40 (sum of $l_8 = 0$ and $l_9 = 40$). The second on machine M_4, with a setup s_3, and operations O_{15} and O_7 with a total load of 80 (sum of $l_{15} = 20$ and $l_7 = 60$). Both occupations respects the capacity of machines M_2 and M_4, with $q_2 = 80$ and $q_4 = 90$, respectively. It is also possible to note that idleness was created on these machines. On machine M_2, the setup-time of the first batch started only on $t = 9$ due to the non-anticipatory consideration, which must respect the maximum release date between operations O_9 ($r_9 = 9$) and O_8 ($r_8 = 5$). The same happens with machine M_4, where the first batch only started at $t = 7$, equivalent to the release date of operation O_{14} ($r_{14} = 7$). It is possible to observe the completion times of the jobs and the operations that define them. For example, job J_1 was the first to finish, defined by the completion time of operation O_9. This job has the highest weight ($w_1 = 46$). Note that job J_5 has the longest completion time, defined by operation O_7, which can be explained by its low weight ($w_5 = 3$). Operations O_1 and O_2 are only associated with job J_5, and they do not define the completion of the job. They are scheduled in the last batch of machines M_1 and M_2, which means that we can move their batches to start any time that makes them finish before or at the same time as operation O_7, without changing the solution cost.

The eligibility constraint can also be verified. Operations O_8 and O_{15} are assigned to machines M_2 and M_4, as they are the only possible machines for these operations. Note that, in some cases, two batches of the same family are scheduled sequentially with only one operation within each. An example of this happens at the end of machine M_2, where batch containing only operation O_{11} precedes another batch containing only operation O_5, both of family F_1. However, the sum of the load occupation of these operations is 90, with $l_{11} = 30$ and $l_5 = 60$, and thus, it would exceed the capacity of machine M_2 ($q_2 = 80$). The completion times of the jobs on the final solution are $C_1 = 35$, $C_2 = 60$, $C_3 = 68$, $C_4 = 54$ and $C_5 = 90$, with a total weighted completion time ($\sum_{j \in \mathcal{N}} w_j C_j$) of $(35 \times 46) + (60 \times 40) + (68 \times 39) + (54 \times 13) + (90 \times 3) = 7,634$.

3.2.2 Stochastic PLSVSP

Uncertainties are present in all stages of oil exploration and production, including tasks performed by support vessels. In the Stochastic PLSVSP (S-PLSVSP), most of these uncertainties affect the processing times of the pipeline connection operations, changing the company's expectation of its duration. These variations may be caused by several aspects such as climate changes (affecting ocean conditions), the complexity of operations, crew experience, and others. Therefore, instead of considering deterministic processing time p_i for each operation $O_i \in \mathcal{O}$, a random variable P_i with $E[P_i] = p_i$ is used to model stochastic processing times for the operations in order to better deal with the uncertain environment. Based on previous works in the scheduling literature (Juan et al. 2014, González-Neira et al. 2019, Hatami et al. 2018, Villarinho et al. 2021), we use Log-Normal distributions for modeling stochastic processing times. The Log-Normal distribution has two parameters, μ_i and σ_i for each operation $O_i \in \mathcal{O}$, defined by the expressions (3-1) and (3-2), respectively (Juan et al. 2011):

$$\mu_i = \ln(E[P_i]) - \frac{1}{2} \cdot \ln\left(1 + \frac{Var[P_i]}{E[P_i]^2}\right), \tag{3-1}$$

$$\sigma_i = \left|\sqrt{\ln\left(1 + \frac{Var[P_i]}{E[P_i]^2}\right)}\right|. \tag{3-2}$$

Following the approach proposed by Juan et al. (2011), we define $Var[P_i] = \delta_p \cdot E[P_i]$, where $\delta_p > 0$ is the processing time variance parameter, used to create different levels of processing times uncertainty.

Figure 3.6 depicts the Log-Normal distribution for a random variable P_i with the expected processing time $E[P_i] = 15$, considering three scenarios of uncertainty (low, medium, and high), with the following values for the variance parameter: $\delta_p \in \{0.5, 2.0, 5.0\}$. Note that, in the low-variance scenario, the curve is more narrowed, generating values closer to $E[P_i]$. For higher values on δ_p, the curve gets a more spread shape, increasing the uncertainty on the processing times of operation O_i, enlarging the range of possible values.

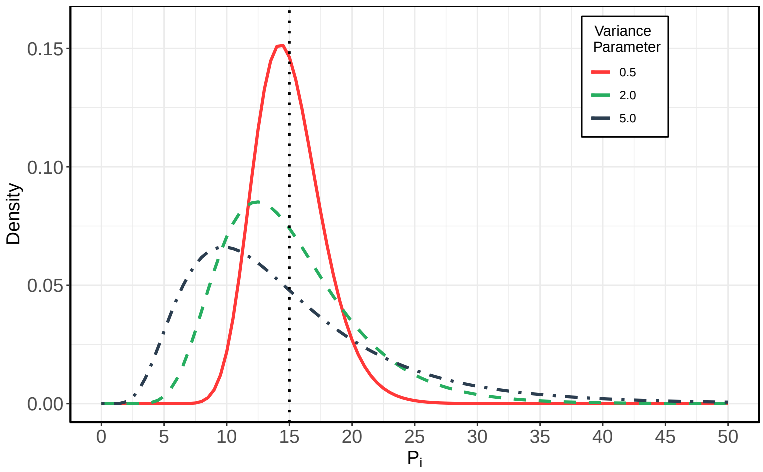


Figure 3.6: Log-Normal distributions for the P_i random variable, with $E[P_i] = 15$, within different uncertainty scenarios.

Another relevant aspect of uncertainty in the PLSVSP concerns the arrival of pipelines at the port (release date of operations). Each pipeline is specially designed for a specific well based on the water depth, total distance to the platform, and other characteristics of the well. When the company orders a new pipeline, a delivery date is defined with the producer and considered by the company as the release date r_i for the operation O_i of the ordered pipeline. However, uncertainties in the production process, pipeline specifications, transportation logistics, and others can affect the defined release dates. Thus, in the Stochastic PLSVSP we assume that the release dates for each operation $O_i \in \mathcal{O}$ is a random variable R_i with $E[R_i] = r_i$. Again, we use Log-Normal distributions to model the stochastic release dates, following the same rule defined for modeling the stochastic processing times. And, we define $Var[R_i] = \delta_r \cdot E[R_i]$, where $\delta_r > 0$ is the release date variance parameter.

4
Mathematical Formulations for the PLSVSP

In this chapter, we propose three mathematical formulations for the PLSV scheduling problem. We first introduce a positional scheduling formulation followed by a time-indexed formulation. Both models extend the parallel machine scheduling formulations described by Unlu and Mason (2010). The third formulation adapts a parallel batching machine scheduling formulation proposed by Ham et al. (2017), using a dispatching rule to sequence operations inside batches. Appendix B.1 contains the definitions of the Sets, Parameters, and Variables used in the proposed formulations.

4.0.1
Positional Scheduling Formulation

The first formulation uses positions $p \in \mathcal{P}$ to define the sequence of setup times and operations on the machines. A setup marks the start of a batch (setting the occupation to zero). We use $\mathcal{P}_k$ to limit the number of available positions on each machine M_k, considering the eligibility constraint. Thus, for each machine, the number of available positions is equal to twice the total number of eligible operations. We do this to allow each machine to process its entire subset of eligible operations in single batches (i.e., batches with only one operation inside). The positional formulation uses a binary variable X_{ik}^{p} which is 1 if operation O_i is scheduled as the p-th operation of machine M_k, a binary variable Y_{gk}^{p} which is 1 if the p-th position of machine M_k is a setup time of family F_g. The continuous variables S_k^p represent the starting of position p in machine M_k. As mentioned in Chapter 3, variables C_i and C_j represent the completion time of operations and jobs, respectively. This formulation expands the one presented in Unlu and Mason (2010), created to solve a parallel machine scheduling problem. We added two continuous variables to deal with the capacity and with the non-anticipatory setup times. The first is defined by L_k^p and represents the total cumulative load in position p, on machine M_k. The second is defined by R_k^p and represents the release time in position p, on machine M_k, looking ahead to all scheduled operations until a new scheduled setup is found. The proposed formulation is described as follows:

$$\min \sum_{j \in \mathcal{N}} w_j C_j \tag{4-1}$$

subject to

$$\sum_{k \in \mathcal{M}_i} \sum_{p \in \mathcal{P}_k} X_{ik}^p = 1 \qquad \forall i \in \mathcal{O} \tag{4-2}$$

$$\sum_{i \in \mathcal{O}} X_{ik}^p + \sum_{g \in \mathcal{F}} Y_{gk}^p \leq 1 \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-3}$$

$$L_k^p \geq L_k^{(p-1)} + \sum_{i \in \mathcal{O}} l_i X_{ik}^p - \sum_{f \in \mathcal{F}} q_k Y_{gk}^p \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \mid p > 0 \tag{4-4}$$

$$L_k^p \leq q_k \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-5}$$

$$\sum_{i \in \mathcal{O}_g} X_{ik}^p \leq \sum_{i \in \mathcal{O}_g} X_{ik}^{(p-1)} + Y_{gk}^{(p-1)} \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \mid p > 0, g \in \mathcal{F} \tag{4-6}$$

$$R_k^p \geq \sum_{i \in \mathcal{O}} r_i X_{ik}^{(p+1)} \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-7}$$

$$R_k^p \geq R_k^{(p+1)} - \sum_{g \in \mathcal{F}} r_{max} Y_{gk}^{(p+1)} \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-8}$$

$$S_k^p \geq S_k^{(p-1)} + \sum_{i \in \mathcal{O}} p_i X_{ik}^{(p-1)} + \sum_{g \in \mathcal{F}} s_g Y_{gk}^{(p-1)} \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \mid p > 0 \tag{4-9}$$

$$S_k^p \geq r_k \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-10}$$

$$S_k^p \geq R_k^p \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-11}$$

$$C_i \geq S_k^p + p_i - (1 - X_{ik}^p) M \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, p \in \mathcal{P}_k \tag{4-12}$$

$$C_j \geq C_i \qquad \forall j \in \mathcal{N}, i \in \mathcal{O}_j \tag{4-13}$$

$$X_{ik}^p \in \{0, 1\} \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, p \in \mathcal{P}_k \mid p > 0 \tag{4-14}$$

$$Y_{gk}^p \in \{0, 1\} \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k, g \in \mathcal{F} \tag{4-15}$$

$$C_i \geq 0 \qquad \forall i \in \mathcal{O} \tag{4-16}$$

$$L_k^p, S_k^p, P_k^p \geq 0 \qquad \forall k \in \mathcal{M}, p \in \mathcal{P}_k \tag{4-17}$$

The objective function (4-1) minimizes the weighted completion time of the jobs. Constraints (4-2) states each operations must be executed exactly once. Constraints (4-3) ensure that each machine processes only one operation or a setup per position. Constraints (4-4) calculate the cumulative load of a position as the sum of the load of the previous position plus the load of the operation assigned to the position. If one setup time is assigned to this position, the cumulative load is equals to zero. Constraints (4-5) guarantee that the cumulative load is less than or equal to the maximum load limit on each machine. Constraints (4-6) ensure that a given operation of a family can only be scheduled after another operation or a setup time of the same family. Constraints (4-7) compute the release of a position from the release date of the operation assigned to the subsequent position on each machine. Constraints (4-8) also compute the release of a position, ensuring that it is greater or equal to the release of the following position on each machine. If there is a setup assigned in the succeeding position, the release variable will be null because of the r_{max} parameter, computed as $r_{max} = \max_{O_i \in \mathcal{O}}\{r_i\}$. Constraints (4-9) calculate the start time of each position on each machine as the start time of the previous position plus the processing time of the operation or setup assigned to the previous position. Constraints (4-10) ensure that the start time of each position must respect the release date of the machine to which it is assigned. Constraints (4-11) force the start of each position on each machine to respect the calculated release time of the position. Constraints (4-12) determine operations completion times based on the assigned position start time and the operation processing time. We use a large number M to relax this constraint for positions on machines to which an operation has not been assigned. The value of M is given by the maximum completion time among all operations, supposing that all of them are scheduled in the same machine in individual batches, with the first batch starting its processing on the largest release date of the operations. Constraints (4-13) ensure that the completion time of a job is the maximum completion time among the operations that comprise it. Finally, Constraints (4-14)-(4-17) present the variables' domains.

4.0.2 Time-Index Scheduling Formulation

In time-indexed formulations, the horizon consists of discrete periods, where each period $t \in \mathcal{T}$ represents a continuous interval starting at time $t-1$ and ending at time t. We limit the available periods $\mathcal{T}_k$ for each machine M_k by taking into account the processing times, release dates and family setup times of the eligible operations subset, and the machine release date. Let $\mathcal{O}_k$ be the subsets of operations that a machine M_k is eligible to execute. Thus, for each machine M_k, the horizon is limited by $t \geq \max\{\min_{O_i \in \mathcal{O}_k}\{r_i\},\ r_k\}$ and $t \leq \max\{\max_{O_i \in \mathcal{O}_k}\{r_i\},\ r_k\} + \sum_{O_i \in \mathcal{O}_k}(s_{f_i} + p_i)$. This formulation uses a binary variable X_{ik}^t which is 1 if operation O_i starts its processing in period t on machine M_k and a binary variable Y_{gk}^t which is 1 if a setup time of family F_g starts in period t in machine M_k. Variables C_i and C_j represent the completion time of operations and jobs, respectively. We also extended the formulation from a parallel machine scheduling one, presented in Unlu and Mason (2010). In the same way, as in the positional formulation, two continuous variables are considered to deal with the capacity and with the non-anticipatory setup times. The first is defined by L_k^t and represents the total load from the last setup up to period t. The second is defined by R_k^t and represents the release time in period t, looking ahead to all scheduled operations until a new scheduled setup is found. The proposed formulation is described as follows:

$$\min \sum_{j \in \mathcal{N}} w_j C_j \tag{4-18}$$

subject to

$$\sum_{k \in \mathcal{M}_i} \sum_{t \in \mathcal{T}_k} X_{ik}^t = 1 \qquad \forall i \in \mathcal{O} \tag{4-19}$$

$$\sum_{i \in \mathcal{O}} \sum_{t'=max\{0,t-p_i+1\}}^{t} X_{ik}^{t'} + \sum_{g \in \mathcal{F}} \sum_{t'=max\{0,t-s_g+1\}}^{t} Y_{gk}^{t'} \leq 1 \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-20}$$

$$L_k^t \geq L_k^{(t-1)} + \sum_{i \in \mathcal{O}} l_i X_{ik}^t - \sum_{g \in \mathcal{F}} q_k Y_{gk}^t \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \mid t > 0 \tag{4-21}$$

$$L_k^t \leq q_k \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-22}$$

$$\sum_{i \in \mathcal{O}_f} X_{ik}^t \leq \sum_{i \in \mathcal{O}_f} X_{ik}^{\max\{0,t-p_i\}} + Y_{gk}^{\max\{0,t-s_f\}} \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k, g \in \mathcal{F} \tag{4-23}$$

$$R_k^t \geq \sum_{i \in \mathcal{O}} r_i X_{ik}^t \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-24}$$

$$R_k^t \geq R_k^{(t+1)} - \sum_{g \in \mathcal{F}} r_{max} Y_{gk}^{(t+1)} \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-25}$$

$$\sum_{g \in \mathcal{F}} (t) Y_{gk}^t \geq R_k^t - \left(1 - \sum_{g \in \mathcal{F}} Y_{gk}^t\right) r_{max} \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-26}$$

$$C_i = \sum_{k \in \mathcal{M}} \sum_{t \in \mathcal{T}_k} (t + p_i - 1) X_{ik}^t \qquad \forall i \in \mathcal{O} \tag{4-27}$$

$$C_j \geq C_i \qquad \forall j \in \mathcal{N}, i \in \mathcal{O}_j \tag{4-28}$$

$$X_{ik}^t \in \{0,1\} \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, t \in \mathcal{T}_k \tag{4-29}$$

$$Y_{gk}^t \in \{0,1\} \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k, g \in \mathcal{F} \tag{4-30}$$

$$C_i \geq 0 \qquad \forall i \in \mathcal{O} \tag{4-31}$$

$$L_k^t, R_k^t \geq 0 \qquad \forall k \in \mathcal{M}, t \in \mathcal{T}_k \tag{4-32}$$

The objective function (4-18) minimizes the weighted completion time of the jobs. Constraints (4-19) state each job must be executed exactly once.

Constraints (4-20) ensure the non overlap of operations and setups. Constraints (4-21) calculate the cumulative load of a period as the sum of the load of the previous period plus the load of the operation starting its processing at this period. If a setup starts at this period, the cumulative load will be zero. Constraints (4-22) guarantee that the cumulative load in each period is less than or equal to the machine capacity. Constraints (4-23) ensure that one operation of a family can only be scheduled after another operation or a setup time of the same family. Constraints (4-24) compute the release of a period from the release date of the operation starting at this period. Constraints (4-25) also compute the release of a period, ensuring that it is greater or equal to the release of the subsequent period on each machine. If there is a setup assigned in the succeeding period, the release variable will tend to zero. Constraints (4-26) force the start of each setup on each machine to respect the calculated release variable of the period. Constraints (4-27) calculate the completion time of each operation as the sum of the start time of the operation plus its duration. Constraints (4-28) ensure that the completion time of a job is the maximum completion time among the operations that comprise it. Finally, Constraints (4-29)-(4-32) present the variables' domains. To accomplish the operations release dates, we generate the allocation variables X_{ik}^t only for periods $t \geq r_i + s_{f_i} + 1$.

4.0.3 Batch Scheduling Formulation

In this formulation, we adapted a parallel batching machine model, presented by Ham et al. (2017). In batching machine formulations, operations are processed in parallel with the total processing time of a batch defined as the maximum processing time of the operations contained in it. However, we consider that operations are processed sequentially, and a batch's processing time is given by the sum of all processing times of its operations. To achieve this, we use a dispatching rule to sequence operations within batches. We limit the number of available batches for each machine M_k, according to the total number of eligible operations on each of them, and create the set $\mathcal{B}_k$. The model decides which operations will be allocated in each batch.

We use the Weighted Shortest Processing Time (WSPT) dispatching rule (Pinedo 2012) to define the sequence of the operations inside batches. Since weights refer to jobs rather than operations, we define relative weights $w_i = \max_{J_j \in \mathcal{N}_i} \{w_j\}$ for operations. Other rules are used to ensure complete ordering of all possible combinations of operations within batches. The model uses the subsets $\mathcal{O}_i$, shown in Equation (4-33), to identify for each operation O_i, the subsets of operations $O_{\hat{\imath}} \in \mathcal{O}_i$ that will precede operation O_i if they are in the same batch. Using the WSPT rule does not guarantee that, even getting a zero-gap when solving this formulation, we will achieve an optimal solution.

$$\mathcal{O}_i = \left\{ O_{\hat{\imath}} \in \mathcal{O} \;\middle|\; \left(\frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} > \frac{w_i}{p_i}\right) \vee \left(\frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} = \frac{w_i}{p_i} \wedge w_{\hat{\imath}} > w_i\right) \vee \left(\frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} = \frac{w_i}{p_i} \wedge w_{\hat{\imath}} = w_i \wedge O_{\hat{\imath}} < O_i\right) \right\} \quad \forall O_i \in \mathcal{O} \tag{4-33}$$

The serial batching formulation uses a binary variable X_{ik}^b which is 1 if operation O_i is scheduled in the b-th batch of machine M_k, a binary variable Y_{fk}^b which is 1 if the b-th batch of machine M_k is of family F_g. The continuous variables S_k^b and P_k^b represent the starting time and the total processing time of the b-th batch on machine k, respectively. The continuous variables C_i and C_j represent the completion times of operation O_i and job J_j, respectively. The mathematical formulation is as follows:

$$\min \sum_{j \in \mathcal{N}} w_j C_j \tag{4-34}$$

subject to

$$\sum_{k \in \mathcal{M}_i} \sum_{b \in \mathcal{B}_k} X_{ik}^b = 1 \qquad \forall i \in \mathcal{O} \tag{4-35}$$

$$\sum_{g \in \mathcal{F}} Y_{gk}^b \leq 1 \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \tag{4-36}$$

$$X_{ik}^b \leq Y_{f_i k}^b \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, b \in \mathcal{B}_k \tag{4-37}$$

$$\sum_{i \in \mathcal{O}} l_i X_{ik}^b \leq q_k \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \tag{4-38}$$

$$P_k^b \geq \sum_{i \in \mathcal{O}} p_i X_{ik}^b + \sum_{g \in \mathcal{F}} s_f Y_{gk}^b \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \tag{4-39}$$

$$S_k^b \geq r_k \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \tag{4-40}$$

$$S_k^{b+1} \geq S_k^b + P_k^b \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \tag{4-41}$$

$$S_k^b \geq r_i X_{ik}^b \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, b \in \mathcal{B}_k \tag{4-42}$$

$$C_i \geq S_k^b + p_i + s_{f_i} + \sum_{\hat{\imath} \in \mathcal{O}_i} p_{\hat{\imath}} X_{\hat{\imath}k}^b - \left(1 - X_{ik}^b\right) M \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, b \in \mathcal{B}_k \tag{4-43}$$

$$C_j \geq C_i \qquad \forall j \in \mathcal{N}, i \in \mathcal{O}_j \tag{4-44}$$

$$X_{ik}^b \in \{0,1\} \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, b \in \mathcal{B}_k \tag{4-45}$$

$$Y_{gk}^b \in \{0,1\} \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k, g \in \mathcal{F} \tag{4-46}$$

$$C_i \geq 0 \qquad \forall i \in \mathcal{O} \qquad (4\text{-}47)$$

$$S_k^b, P_k^b \geq 0 \qquad \forall k \in \mathcal{M}, b \in \mathcal{B}_k \qquad (4\text{-}48)$$

The objective function (4-34) minimizes the weighted completion of the jobs. Constraints (4-35) state that each operation is executed exactly once. Constraints (4-36) allow only at most one family to be associated with each batch. Constraints (4-37) assure operations can only be scheduled in a given machine and batch using the selected family. Constraints (4-38) control the occupation of each batch, avoiding the violation of the machine capacity. Constraints (4-39) set the total processing time of a batch as the total processing time of the operations it contains plus the setup time of the chosen family. Constraints (4-40)-(4-42) control the starting time of the batches. Constraints (4-40) set them as at least the machine's release date. Constraints (4-41) ensure each batch starts after the processing time of the previous batch. Constraints (4-42) force batch starts to respect the maximum release date between the operations scheduled in it (i.e., they guarantee the non-anticipation). Constraints (4-43) calculate the completion time of one operation as the sum of the processing time of all operations scheduled before, the setup time of the batch family, its own processing time and the starting time of the batch, only for the scheduled batch, family and machine. A large number M is used to relax this constraint for batches on machines to which operations are not assigned. Constraints (4-44) ensure that the completion time of a job is the maximum completion time among the operations that comprise it. Finally, Constraints (4-45)-(4-48) present the variables' domains.

4.1 Computational Experiments

In this section, we present and discuss the results of the computational experiments on the three proposed formulations running within the same conditions. To test the performance of the formulations, we generated a set of PLSV scheduling instances based on real data, from a Brazilian company, related to the pre-salt basin. We performed the experiments on a machine with an Intel i7-3960 CPU at 3.3GHz and 64 GB of RAM. The ILP formulations were implemented in AIMMS 4.4 and solved by GUROBI 7.5 solver running in 4 threads within a 6 hours time limit. This time-limit has been set with the company to fit the way the planning is done. In general, the company optimizes a schedule for the coming months without the need for an immediate answer. However, when a response is required on the same business day, the time-limit still works. We refer as `Positional`, `Time-Index` and `Batch` for the

Positional Scheduling Formulation (Section 4.0.1), the *Time-Index Scheduling Formulation* (Section 4.0.2) and the *Batch Scheduling Formulation* (Section 4.0.3), respectively.

4.1.1 Data Generation

The studied company provided a set of historical data with several PLSV schedules, containing the voyages planned on each vessel and details on the pipeline connections, indicating their associated wells. PLSV planners shared more information about vessel eligibility and capacity, pipelines occupation, and well production potential during meetings held to understand the process. Based on this, we developed a set of PLSV instances to test our formulations. We defined the number of machines and operations as $m = \{2, 4\}$ and $o = \{15, 25, 50\}$, respectively. The number of jobs was defined as $n = \lfloor o/3 \rfloor$ and the number of families was set as 3, as described in Table 4.1. To generate different ranges for the release dates, we defined a factor $\alpha = \{0.25, 0.50, 0.75\}$. Smaller values for this factor generate release dates for the operations and machines closest to zero. If set as zero, release dates are disregarded in the problem, i.e., $r_i = 0$ and $r_k = 0$ for all operations and machines, respectively. The factor $\beta = \{0.7, 0.9\}$ enables the generation of instances with different machines eligibility. Higher values for this factor represent a greater probability of a machine being eligible to execute some operation. If set as one, all machines will be eligible to execute all operations. A factor $\gamma = \{0.05, 0.15\}$ defines the probability of associating an operation to a job during the instance generation. Higher values increase the intersections between different sets of operations. If set to zero, no intersection is allowed.

All combinations among the proposed factors were considered, generating 72 instances. Before generating the instances, a pre-processing step creates part of the data used in the definition of some parameters. The parameters generated on this step, the input data, and the parameters of the final instance are described in Table 4.1. We used $U(a, b)$ to define a continuous uniform distribution between a and b and $U\{a, b, c\}$ for a discrete uniform distribution between a and b with step size c. The developed research data is available online Abu-Marrul et al. (2019).

4.1.2 Results

In the following analysis, we evaluate the solution quality of each formulation in terms of the relative optimality gap of the obtained solutions

Table 4.1: Settings in randomly generated instances of the PLSV scheduling problem.

Generation step	**Parameter**	**Value, range or distributions**
Input problem sets	Number of operations	$o = \{15, 25, 50\}$
	Number of machines	$m = \{4, 8\}$
	Number of jobs	$n = \lfloor o/3 \rfloor$
	Number of families	$f = 3$
Input factor sets	Release date factor	$\alpha = \{0.25, 0.5, 0.75\}$
	Eligibility factor	$\beta = \{0.7, 0.9\}$
	Association factor	$\gamma = \{0.05, 0.15\}$
Pre-processing	Operations $\times$ Jobs coefficient	$\mathcal{ON}_{ij} \sim U(0, 1)$
	Operations $\times$ Machines coefficient	$\mathcal{OM}_{ik} \sim U(0, 1)$
	Maximum release date	$MR = \left\lceil \alpha \times \frac{\sum_{O_i \in \mathcal{O}} (p_i + s_{f_i})}{m} \right\rceil$
Instance parameters	Operations processing times	$p_i \sim U\{1, 30, 1\}$
	Operations families	$f_i \sim U\{1, f, 1\}$
	Operations load occupation or size	$l_i \sim U\{0, 100, 10\}$
	Operations release date	$r_i \sim U\{0, MR, 1\}$
	Operations $\times$ Jobs subsets	$\mathcal{O}_j = \{O_i \in \mathcal{O} \mid \mathcal{ON}_{ij} \leq \gamma\}$
	Jobs weight	$w_j \sim U\{1, 50, 1\}$
	Machines capacity	$q_k \sim U\{80, 100, 10\}$
	Families setup duration	$s_g \sim U\{5, 10, 1\}$
	Machines release date	$r_k \sim U\{0, MR, 1\}$
	Eligibility Subsets	$\mathcal{M}_i = \{M_k \in \mathcal{M} \mid (\mathcal{OM}_{ik} \leq \beta) \text{ and } (l_i \leq q_k)\}$

calculated using the lower bounds found when solving `Time-Index`. In Appendix B.2, we present a comparison between lower bounds, showing that `Time-Index` generates the best lower bounds for all tested instances. We report the relative optimality gaps in the way the GUROBI solver does, as $|ObjBound - ObjVal|/ObjVal \times 100$, where, in our case, $ObjVal$ and $ObjBound$ are the generated solutions and `Time-Index` lower bounds, respectively. Table 4.2 shows the relative optimality gaps, to `Time-Index` lower bounds, for each formulation per instance group. We consider a group as a combination of the number of operations and the number of machines. Information on the number of operations (o) and the number of machines (m) are shown in the first two columns of the table. With this grouping scheme, we get six groups with 12 instances on each. Furthermore, for each formulation, we show the minimum percentage gap (min), the average percentage gap (avg), the maximum percentage gap (max), the standard deviation between the gaps (sd), the number of optimal solutions found (opt) and how many times each formulation has found the best solution (best) for the instances of each group. Finally, the last row depicts the same results for all instances.

Note that for groups 15–4 and 15–8, `Time-Index` results are optimal for 23 out of 24 instances and the best for all 24 with a worst-case gap of 0.01%. The other formulations also performed well on these groups, with an average gap below 1%. Regarding groups 25–4 and 25–8, we observe that `Time-Index`

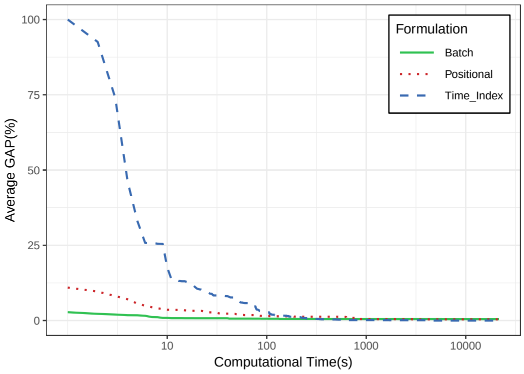
Table 4.2: General results per instance group.

o	m	Positional						Time-Index						Batch					
		min	avg	max	sd	opt	best	min	avg	max	sd	opt	best	min	avg	max	sd	opt	best
15	4	0.00	0.40	3.26	0.92	7	7	0.00	0.00	0.01	0.00	11	12	0.00	0.46	1.59	0.65	6	7
	8	0.00	0.09	0.40	0.15	8	8	0.00	0.00	0.00	0.00	12	12	0.00	0.02	0.22	0.06	11	11
25	4	0.31	6.63	12.30	4.00	0	4	0.00	6.65	12.44	4.27	1	6	1.00	6.43	13.47	3.90	0	3
	8	0.18	5.08	11.89	3.32	0	1	0.00	3.69	8.04	2.80	1	10	0.28	4.70	9.68	3.45	0	2
50	4	9.07	15.62	24.04	5.64	0	1	10.93	27.77	42.20	9.71	0	0	6.76	13.31	20.28	4.93	0	11
	8	8.14	16.72	26.36	6.07	0	1	10.75	27.13	41.48	10.19	0	0	5.70	15.04	26.78	6.65	0	11
All		0.00	7.42	26.36	7.71	15	22	0.00	10.87	42.20	13.39	25	40	0.00	6.66	26.78	6.99	17	45

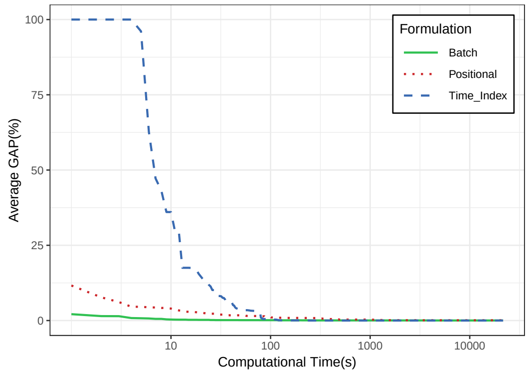
found only one optimal solution for each group. Besides, one can see that the average gaps are similar for all formulations. Note that when the number of operations increases, the gaps also increases, with `Time-Index` being the most affected, reaching maximum gaps above 41% for groups 50–4 and 50–8. For these groups, `Batch` performed better with the smallest gaps on average (13.33% and 15.04%). Note that `Batch` also found the best solution for 22 of the 24 instances with 50 operations. Considering the complete set of instances, one notes that `Batch` also performed better, finding the best solutions on 45 out of 72 instances with a smaller average gap. An analysis of the number of variables and constraints generated by each model is presented in Appendix B.3, showing that `Batch` generates, on average, 95.39% fewer variables than `Time-Index` and 51.37% than `Positional`. Regarding the constraints, these numbers are 71.81% and 5.56%, respectively, which explains the good performance of `Batch`.

Since the problem is related to a real-world scheduling demand, it is crucial to know how solution quality evolves during solution time for practical purposes. Based on this assumption, Figure 4.1 shows an analysis of the average gap evolution through the computational time, in seconds, for each group. Note that the behavior is similar in all groups.

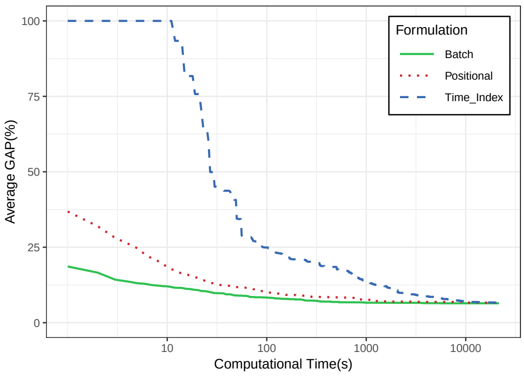
Figures 4.1a and 4.1b show the evolution of the average gap on groups 15–4 and 15–8. It is possible to observe that `Time-Index` achieved the best average gaps when compared to `Positional` and `Batch`, in less than 350 seconds on both groups. However, we can see that the results of all formulations are competitive with each other in about 100 seconds in these groups. Figures 4.1c and 4.1d show the average gap evolution for groups 25–4 and 25–8, respectively. Note that `Batch` dominates the other formulations in group 25–4 with the best average gap during complete solving. In group 25-8, `Batch` is also better than `Positional` but was overcome by `Time-Index` results after 2300 seconds. In Figures 4.1e and 4.1f, the evolution of the average gaps are shown for groups 50–4 and 50–8. In these groups, `Time-Index` could not compete with the other two proposed formulations. Note that, `Batch` dominates `Positional` and `Time-Index` during complete solving, with smaller average gaps in booth



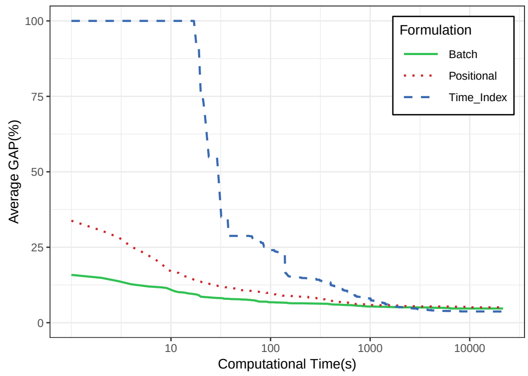
(a) 15 Operations and 4 Machines.



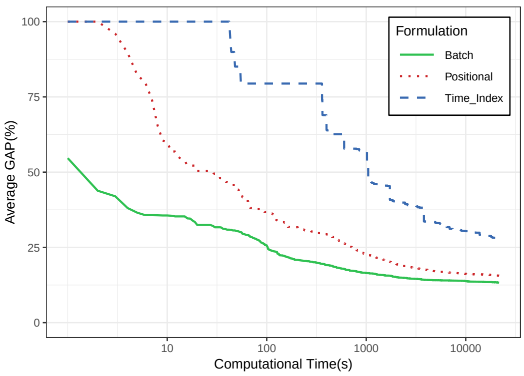
(b) 15 Operations and 8 Machines.



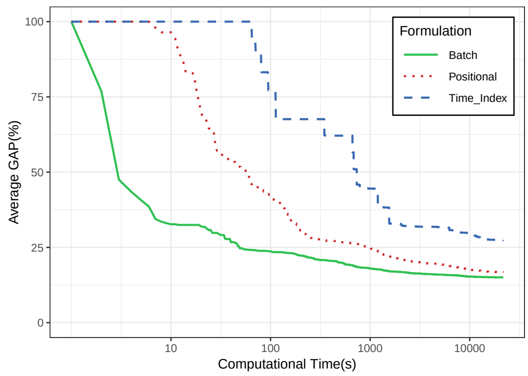
(c) 25 Operations and 4 Machines.



(d) 25 Operations and 8 Machines.



(e) 50 Operations and 4 Machines.



(f) 50 Operations and 8 Machines.

Figure 4.1: Evolution of the average gap in relation to the lower bound between formulations.

groups (50-4 and 50-8).

The same visualization is shown in Figure 4.2, with the average gap evolution through the computational time, in seconds, for all instances. The computational time axis was defined on a $\log_{10}$ scale for better visualization. This approach also uses the lower bounds generated by `Time-Index` to compute gaps. Note that `Batch` approach dominates `Time-Index` and `Positional` during the entire run with the smallest final average gap among the formulations.

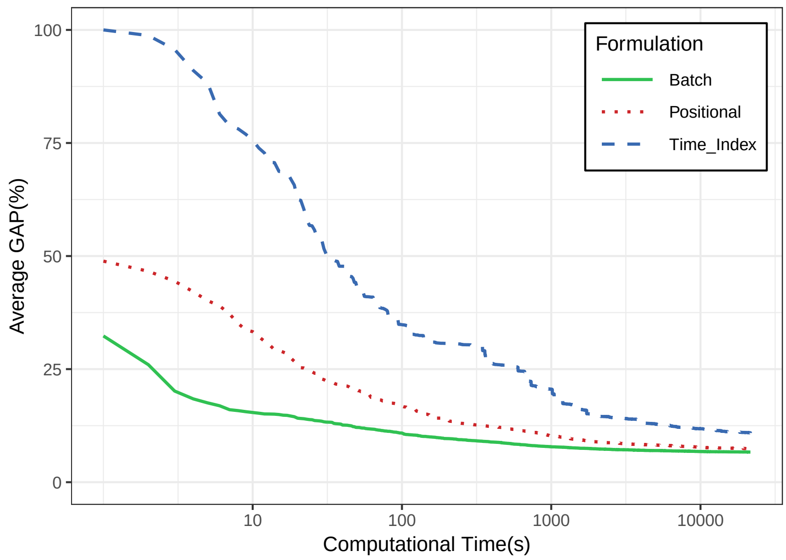


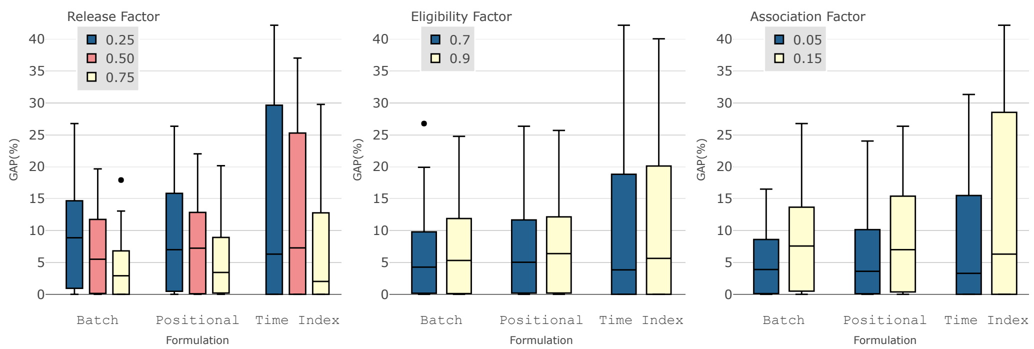
Figure 4.2: Evolution of the average gap in relation to the lower bound for all instances.

4.1.3 Sensitivity analysis

In order to obtain evidence about the impact of the input factors used in the instance generation process described in Section 4.1.1, Figure 4.3 depicts the distribution of the gaps for each factor type. All gaps were computed using `Time-Index` lower bounds.

Figure 4.3a shows the distribution for each release factor (α). Note that higher values on this factor result in smaller gaps for all formulations. Since we are focusing on minimizing a function based on the operations completion times, more spread release dates make it easy to generate bounds for the problem. Figure 4.3b depicts the same analysis for the eligibility factors (β), showing that smaller values on this factor generate better solutions. Smaller factors, in that case, means that there will be fewer eligible operations for each machine. The distribution of the gaps for each association factor (γ) is shown in Figure 4.3c. Note that higher factors generate larger gaps. As discussed in

Chapter 3, this is the main realistic feature of the PLSV scheduling problem, where we have operations associated with several jobs simultaneously. This characteristic directly impacts the completion times since, to be completed, a job needs to have all of its operations completed. When more associations are generated, more intersection will exist among the operations sets, the completion time of one operation may affect the completion time of multiple jobs with different weights. Note that this condition makes the problem more challenging to solve and, therefore, a relevant feature to address.



(a) Release Factor. (b) Eligibility Factor. (c) Association Factor.

Figure 4.3: Boxplot of the relative gap to Time-Index lower bounds for each factor type.

4.2
Discussion

In this section, we developed three ILP formulations to tackle the PLSVSP. The first is a positional scheduling model, which generates some available positions on each machine and decide how to schedule operations in these positions. The second uses a time-indexed formulation to schedule the operations in a set of discrete periods. The third is a batching machine scheduling formulation using a WSPT dispatching rule to sequence operations within batches.

The batch formulation showed better results on a set of 72 instances generated from real data of the studied company. The model reached an average gap of 6.66%, against 7.42% achieved by the positional and 10.87% by the time-indexed one. This formulation also proved to be the best approach for faster solutions and larger instances. On instances with 50 operations to schedule, the batch formulation dominates the other two, finding the best solution 22 times out of the 24 tested instances with this number of operations to schedule. All solutions were evaluated in comparison to the lower bounds generated by the time-indexed model, which presented the best lower bounds for all instances.

Computational experiments have shown that when the intersection between operations sets increases, the gaps obtained also increase. This behavior is an indicator of the importance of considering this aspect, being a real feature of the problem, and contributing to the machine scheduling literature by creating a more challenging problem to solve. Using a dispatching rule within a mathematical formulation has proved to be a beneficial approach, raising some interesting questions about its application on more classic family-scheduling problems for future work. Depending on the objective function, different dispatching rules may be considered and compared.

The results were presented and validated with the studied company. A decision support system designed to assist in the company's PLSV fleet scheduling process is under development. The system has an optimization module in which the batch formulation, developed in this study, was incorporated. The first experiments using the optimization module were carried out during the tactical planning of the year 2020 for the Brazilian pre-salt basin demand. The schedules generated by the optimization module were considered by the company when defining the actual plan for the PLSV fleet. The experiments showed the potential of the approach in solving such an important problem by providing good quality solutions used as decision support by the company. In the future, the company expects to have a fully functional system capable of assisting the scheduling process of its entire PLSV fleet and including other exploratory basins in the Brazilian offshore region.

5
Constructive Heuristics for the PLSVSP

In this chapter, we present several constructive heuristics to solve the PLSVSP, using scheduling dispatching rules, and defining how to do the machine assignment and to construct batches.

All heuristics described here creates batches by assigning operations sequentially to the machines. Therefore, the chosen method selects the next operation to schedule and assigns a machine to the operation at each iteration. Then, the method decides whether to insert the selected operation on the last batch (called current batch) or to create a new batch to insert it, on the assigned machine. If a new batch is created, the selected operation is sequenced as the first one inside the new batch, i.e., after a new family setup time also inserted in the machine schedule. Otherwise, the operation is scheduled as the last one in the current batch on the assigned machine. There are three situations that force the creation of a new batch on the selected machine: (1) when the machine is empty; (2) when the current batch on the machine is of a different family from the selected operation; (3) when the insertion of the operation in the current batch exceeds the machine capacity.

Initially, we present a general construction procedure without detailing the steps of operations and machine selection. In the sequence, we show two different ways to choose operations and machines. In the first, operations are selected according to a chosen rule, and then a machine is assigned to perform it. In the other, the procedure selects operation-machine pairs, choosing the operation and the machine simultaneously. Table 5.1 shows the new variables and sets used to store information about the solution during heuristic procedures.

5.0.1
General Constructive Procedure

In this section, we present the general procedure (Algorithm 1) used to construct solutions for the PLSVSP heuristically. The procedure returns a list of schedules $\sigma = (\sigma_1, \ldots, \sigma_{|\mathcal{M}|})$ for the machines $M_k \in \mathcal{M}$. Each schedule σ_k

Table 5.1: Variables and Sets used in the constructive heuristics.

Type	Name	Description
Set	$\mathcal{U}$	Subset of unscheduled operations ($\mathcal{U} \subseteq \mathcal{O}$). In the first iteration, $\mathcal{U}$ is equal to the set $\mathcal{O}$ of operations
Set	$\mathcal{U}_j$	Subset of unscheduled operations associated to job $J_j \in \mathcal{N}$ ($\mathcal{U}_j \subseteq \mathcal{U}$)
Set	$\mathcal{A}_k$	Set of operations scheduled in the current batch on machine $M_k \in \mathcal{M}$
Set	$\mathcal{CB}$	Feasible assignments cb_{ik} of operations $O_i \in \mathcal{O}$ in the current batches of machines $M_k \in \mathcal{M}$ *(to be feasible, the assignment must respect the eligibility, family, and capacity constraints)*
Set	$\mathcal{NB}$	Feasible assignments nb_{ik} of operations $O_i \in \mathcal{O}$ in new batches on machines $M_k \in \mathcal{M}$ *(to be feasible, the assignment must respect the eligibility constraint)*
Variable	C_k	Completion time of machine $M_k \in \mathcal{M}$
Variable	T_i	Minimum completion time among the set M_i of eligible machines for operation $O_i \in \mathcal{O}$. Computed as $T_i = \min_{M_k \in \mathcal{M}_i} \{C_k\}$
Variable	S_k	Starting time of current batch on machine $M_k \in \mathcal{M}$
Variable	L_k	Cumulative load of the current batch on machine $M_k \in \mathcal{M}$
Variable	F_k	Family of the current batch on machine $M_k \in \mathcal{M}$
Variable	Δ_{ik}	Delay at starting the current batch on machine $M_k \in \mathcal{M}$ with the insertion of operation $O_i \in \mathcal{O}$
Variable	C_{ik}^{CB}	Completion time of operation $O_i \in \mathcal{O}$ if inserted in the current batch on machine $M_k \in \mathcal{M}$
Variable	C_{ik}^{NB}	Completion time of operation $O_i \in \mathcal{O}$ if inserted in a new batch on machine $M_k \in \mathcal{M}$
Boolean Variable	*same*	If `true`, the chosen operation is scheduled in the current batch on machine $M_k \in \mathcal{M}$, otherwise, the operation goes to a new batch

contains a sequence of operations and families. A family represents a new setup time, defining the beginning of a batch. At the beginning of the procedure, we estimate weights for the operations (w_i), since weights in the PLSVSP are related to jobs, to be used by the selection procedures, described in the next sections. Five rules for estimating weights are considered, defined in Table 5.2.

The algorithm starts by initializing the variables and sets (Lines 1-3). The method runs until all operations are scheduled (Line 4). The main loop (Lines 4-19) starts by computing the operations weights (w_i) for all unscheduled operations (Line 5). The next operation to schedule and the machine to execute it are selected according to a chosen heuristic (Line 6). The heuristics will be detailed in the nest sections. The value of the variable *same* is also defined

Table 5.2: Rules for estimating operation's weights.

Name	Rule	Description
MAX	$w_i = \max_{J_j \in \mathcal{N}_i} w_j$	Maximum weight of associated wells
SUM	$w_i = \sum_{J_j \in \mathcal{N}_i} w_j$	Sum of the weights of associated wells
AVG	$w_i = \sum_{J_j \in \mathcal{N}_i} \frac{w_j}{\lvert\mathcal{N}_i\rvert}$	Average weight of associated wells
WAVG	$w_i = \sum_{J_j \in \mathcal{N}_i} \frac{w_j}{\lvert\mathcal{O}_j\rvert}$	Weighted average weight of associated wells
WAVGA	$w_i = \sum_{J_j \in \mathcal{N}_i} \frac{w_j}{\lvert\mathcal{U}_j\rvert}$	Weighted average weight of associated wells, considering unscheduled operations

Algorithm 1: General Schedule Construction Procedure

1 $C_k \leftarrow r_k, S_k \leftarrow r_k,\ L_k \leftarrow 0,\ F_k \leftarrow 0,\ \mathcal{A}_k \leftarrow \emptyset, \sigma_k \leftarrow \emptyset\ \forall M_k \in \mathcal{M}$
2 $C_i \leftarrow \infty\ \forall O_i \in \mathcal{O}$
3 $\mathcal{U} \leftarrow \mathcal{O}$
4 **while** $\mathcal{U} \neq \varnothing$ **do**
5   Compute the estimated weights w_i for all operations $O_i \in \mathcal{O}$, following one of the rules defined in Table 5.2
6   Select an operation $O_{i^*} \in \mathcal{U}$ and an eligible machine $M_{k^*} \in \mathcal{M}_{i^*}$, according to a chosen heuristic, also defining the value of *same* as `true` or `false`
7   **if** *same* **then**
8         $\Delta_{i^*k^*} \leftarrow \max(0, r_{i^*} - S_k^*)$
9         $C_{i^*k^*}^{CB} \leftarrow C_k^* + \Delta_{i^*k^*} + p_{i^*}$
10         $S_{k^*} \leftarrow \max(r_{i^*}, S_{k^*}),\ C_{k^*} \leftarrow C_{i^*k^*}^{CB}$
11         $L_{k^*} \leftarrow L_{k^*} + l_{i^*},\ \mathcal{A}_{k^*} \leftarrow \mathcal{A}_{k^*} \cup \{O_{i^*}\}$
12   **else**
13         $C_{i^*k^*}^{NB} \leftarrow \max(r_{i^*}, C_k) + s_{f_{i^*}} + p_{i^*}$
14         $S_{k^*} \leftarrow \max(r_{i^*}, C_{k^*}),\ C_{k^*} \leftarrow C_{i^*k^*}^{NB}$
15         $L_{k^*} \leftarrow l_{i^*},\ \mathcal{A}_{k^*} \leftarrow \{O_{i^*}\}$
16         $\sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{f_{i^*}\}$
17   **end**
18     $\sigma_{k^*} \leftarrow \sigma_{k^*} \cup \{O_{i^*}\},\ F_{k^*} \leftarrow f_{i^*},\ C_{i^*} \leftarrow C_{k^*},\ \mathcal{U} \leftarrow \mathcal{U} \setminus \{O_{i^*}\}$
19 **end**

in this step. If *same* is true, variables and sets regarding the solution are updated (Lines 8-11), considering the insertion of the selected operation O_{i^*} in the current batch of the assigned machine M_{k^*}. Otherwise, these variables and sets are updated in Lines 13-15, considering the insertion of the selected operation O_{i^*} in a new batch on the assigned machine M_{k^*}. Furthermore, the schedule of the chosen machine M_{k^*} is also updated by including a setup time from the family of the selected operation O_{i^*}, defining the beginning of a new batch (Line 16). The schedule and the remaining variables and sets of the algorithm are updated in Line 18. Finally, the procedure returns the final solution σ at Line 20.

The method's main step is defined in the selection component (Line 6), where it decides the next operation to schedule, the machine that will execute the operation, and the creation of a new batch or not. As mentioned earlier, we consider two approaches in this step, one *disjunctive* and one *simultaneous*, described in the following sections.

5.0.2 Operation and Machine Disjunctive Selection Procedure

In the first approach, we first select the next operation to schedule based on a scheduling dispatching rule. Then, we assign it to be executed by an eligible machine, according to the minimum weighted completion time (Algorithm 2). We consider six dispatching rules for this approach. The priority value of π_i indicates the next operation to schedule. At each iteration, the operation O_i with the largest π_i value is selected (Đurasević and Jakobović 2018). We adapt some rules by adding family setup times, and assuming that every operation will be assigned to a new batch. The dispatching rules considered are described in Table 5.3.

Table 5.3: Rules for estimating operation's weights.

Name	Rule	Description
`ERD`	$\pi_i = 1/r_i$	Earliest Release Date
`SPT`	$\pi_i = 1/p_i$	Shortest Processing Time
`LPT`	$\pi_i = p_i$	Longest Processing Time
`MCT`	$\pi_i = \max(T_i, r_i) + p_i + s_{f_i}$	Minimum Completion Time
`WSPT`	$\pi_i = w_i/p_i$	Weighted Shortest Processing Time
`WMCT`	$\pi_i = (\max(T_i, r_i) + p_i + s_{f_i})/w_i$	Weighted Minimum Completion Time

Algorithm 2: Operation and Machine disjunctive Selection

1 $same \leftarrow$ `false`
2 Select the next operation $O_{i^*} \in \mathcal{U}$ to schedule according to a chosen dispatching rule from Table 5.3
3 $\Delta_{i^*k} \leftarrow \max(0, r_{i^*} - S_k)\ \forall M_k \in \mathcal{M}_{i^*}$
4 $C^{CB}_{i^*k} \leftarrow C_k + \Delta_{i^*k} + p_{i^*}\ \forall M_k \in \mathcal{M}_{i^*}$
5 $C^{NB}_{i^*k} \leftarrow \max(r_{i^*}, C_k) + s_{f_{i^*}} + p_{i^*}\ \ \forall M_k \in \mathcal{M}_{i^*}$
6 $\mathcal{CB} \leftarrow \left\{cb_{i^*k} = w_{i^*}C^{CB}_{i^*k} + \sum_{\hat{i}\in\mathcal{B}_k} w_{\hat{i}}\Delta_{i^*k} \mid M_k \in \mathcal{M}_{i^*},\ F_k = f_{i^*},\ L_k + l_{i^*} \leq q_k\right\}$
7 $\mathcal{NB} \leftarrow \left\{nb_{i^*k} = w_{i^*}C^{NB}_{i^*k}\ \mid\ M_k \in \mathcal{M}_{i^*}\right\}$
8 $b_{min} \leftarrow \min\{b : b \in (\mathcal{CB} \cup \mathcal{NB})\}$
9 Select M_{k^*} corresponding to b_{min}
10 **if** $b_{min} \in \mathcal{CB}$ **then**
11     $sane \leftarrow$ `true`
12 **end**
13 **return** $O_{i^*}, M_{k^*}, same$

The algorithm starts by initializing *same* as `false` (Line 1), and selecting the next operation to schedule according to a chosen dispatching rule (Line 2).

The delay at the starting time of the current batch on each eligible machine $M_k \in \mathcal{M}_{i^*}$ is computed in Line 3. In Lines 4 and 5, the selected operation's completion times are computed, considering its insertion in the current batch or in a new batch on the assigned machine, respectively. Next, the method creates sets of feasible assignments (Lines 6 and 7), selecting the assignment with minimum cost (Line 8), identifying the best machine M_{k^*} in Line 9. If the selected element belongs to the set $\mathcal{CB}$, variable *same* is defined as `true` (Line 11). Finally, the procedure returns the selected operation O_{i^*}, the selected machine M_{k^*}, and the variable *same* (Line 13).

5.0.3
Operation and Machine Simultaneous Selection Procedure

The second approach extends one of the heuristics from Weng et al. (2001), by considering the PLSVSP properties, such as the release dates of operations and machines, family setup times, and batch composition, deciding at each iteration the next pair operation/machine simultaneously (Algorithm 3). We call it `WMCT-Pair`.

Algorithm 3: Operation and Machine Simultaneous Selection

1 $\mathit{same} \leftarrow$ `false`
2 $\Delta_{ik} \leftarrow \max(0, r_i - S_k) \ \forall O_i \in \mathcal{O},\ M_k \in \mathcal{M}_{i^*}$
3 $C_{ik}^{CB} \leftarrow C_k + \Delta_{ik} + p_i \ \forall O_i \in \mathcal{O},\ M_k \in \mathcal{M}_{i^*}$
4 $C_{ik}^{NB} \leftarrow \max(r_i, C_k) + s_{f_i} + p_i \ \forall O_i \in \mathcal{O},\ M_k \in \mathcal{M}_{i^*}$
5 $\mathcal{CB} \leftarrow \left\{ cb_{ik} = \frac{C_{ik}^{CB}}{w_i} + \sum_{O_i \in \mathcal{B}_k} \frac{\Delta_{ik}}{w_i} \mid O_i \in \mathcal{U},\ M_k \in \mathcal{M}_i,\ F_k = f_i,\ L_k + l_i \leq q_k \right\}$
6 $\mathcal{NB} \leftarrow \left\{ nb_{ik} = \frac{C_{ik}^{NB}}{w_i} \mid O_i \in \mathcal{U},\ M_k \in \mathcal{M}_i \right\}$
7 $b_{min} \leftarrow \min\{b : b \in (\mathcal{CB} \cup \mathcal{NB})\}$
8 Select O_{i^*} and M_{k^*} corresponding to b_{min}
9 **if** $b_{min} \in \mathcal{CB}$ **then**
10 | $\mathit{sane} \leftarrow$ `true`
11 **end**
12 **return** $O_{i^*}, M_{k^*}, \mathit{same}$

The algorithm starts by initializing *same* as `false` (Line 1). Then, the delay at the starting time of the current batch on each eligible machine $k \in \mathcal{M}_{i^*}$ is computed in Line 2. In Lines 3 and 4, the completion times of all operations are computed, considering its insertion in the current batch or in a new batch on all eligible machines, respectively. Next, the method creates the sets of feasible assignments (Lines 5 and 6), selecting the assignment with minimum cost (Line 7), and identifying the best pair operation/machine in Line 8. If the selected element belongs to the set $\mathcal{CB}$, variable *same* is defined as `true` (Line 10). Finally, the procedure returns the selected operation i^*, the selected machine k^*, and the variable *same* (Line 12).

5.1 Computational Experiments

We introduce in total 19 heuristics, where four do not consider weights, and 15 combine the dispatching rules and the ways of estimating the operations' weights. We tested all of them on a set of 72 PLSVSP instances (Chapter 4) with $m = \{2, 4\}$, and $o = \{15, 25, 50\}$, running the experiments on a computer with 64 GB of RAM and Intel Core i7-8700K CPU of 3.70GHz, using C++ for coding the heuristics and running Linux. The results, in terms of the average relative deviations from the best solutions achieve the heuristics, are presented in Table 5.4. Each instance group, defined by the number of operations and machines, contains 12 instances. The relative deviation is computed as $RD^h_{inst} = TWC^h_{inst}/TWC^{best}_{inst}$, where TWC^h_{inst} is the total weighted completion time of heuristic $h \in \mathcal{H}$ applied to instance $inst \in \mathcal{I}$, and TWC^{best}_{inst} is the best solution obtained for a given instance. The best result for each instance group is shown in bold. All heuristics run in less than 0.1 seconds. Last column ($\#Best$) accounts how many times each heuristic yields the best solution.

Table 5.4: Average deviations from the best solutions.

Heuristic	**Instance Group ($o - m$)**						**All Instances**	**#Best**
	15-4	**15-8**	**25-4**	**25-8**	**50-4**	**50-8**		
`ERD`	1.212	1.198	1.245	1.190	1.261	1.250	1.226	2
`SPT`	1.278	1.217	1.363	1.296	1.442	1.392	1.331	1
`LPT`	1.347	1.153	1.412	1.265	1.471	1.398	1.341	0
`MCT`	1.226	1.165	1.268	1.248	1.254	1.295	1.243	0
`WSPT-MAX`	1.161	1.079	1.224	1.173	1.299	1.255	1.198	1
`WSPT-SUM`	1.156	1.066	1.181	1.141	1.280	1.241	1.178	0
`WSPT-AVG`	1.181	1.085	1.266	1.187	1.327	1.294	1.223	1
`WSPT-WAVG`	1.124	1.076	1.184	1.124	1.234	1.196	1.156	0
`WSPT-WAVGA`	1.085	1.041	1.112	1.067	1.138	1.114	1.093	3
`WMCT-MAX`	1.084	1.040	1.070	1.088	1.046	1.096	1.071	7
`WMCT-SUM`	1.063	1.046	1.093	1.082	1.158	1.132	1.096	3
`WMCT-AVG`	1.101	1.027	1.123	1.118	1.150	1.147	1.111	4
`WMCT-WAVG`	1.065	1.036	1.041	1.059	1.111	1.084	1.066	5
`WMCT-WAVGA`	**1.023**	**1.021**	**1.013**	**1.016**	1.027	**1.003**	**1.017**	34
`WMCT-Pair-MAX`	1.086	1.043	1.063	1.082	1.036	1.091	1.067	9
`WMCT-Pair-SUM`	1.063	1.043	1.087	1.071	1.151	1.124	1.090	4
`WMCT-Pair-AVG`	1.101	1.027	1.123	1.107	1.136	1.136	1.105	2
`WMCT-Pair-WAVG`	1.060	1.035	1.045	1.050	1.100	1.076	1.061	7
`WMCT-Pair-WAVGA`	1.029	1.023	1.024	1.018	**1.026**	1.005	1.021	19

Note that among the heuristics, `WCMT-WAVGA` generated the best average solutions for 5 of 6 groups with the best average deviation of 1.003, achieved on group 50-8. This heuristic also found the highest number of best solutions, on 34 of 72 instances.

5.2 Discussion

In this chapter, we tested 19 heuristics on the set of 72 PSVSP instances (Abu-Marrul et al. 2019). Results show an advantage in terms of solution cost for the `WCMT-WAVGA` heuristic, with an average deviation of 1.017 from the best solutions. These heuristics are useful in practice as they generate reasonable solutions for the PLSVSP without much computational effort.

6
MIP-based Neighborhood Search Matheuristics for the PLSVP

In this chapter, we first provide an extension of the batch scheduling formulation for the PLSVSP, presented in Chapter 4, considering the sequence of operations inside batches as a decision of the model. Then, we introduce two new MIP-based neighborhood searches for batch scheduling formulations, testing its efficiency in an Iterated Local Search (ILS) and a Greedy Randomized Adaptive Search Procedure (GRASP) matheuristic algorithms. The main objectives are to improve the quality of the solution and reduce the computational time compared to running the pure mathematical models.

6.1
Batch Formulation with Sequencing Variables

As stated in Chapter 4, the `Batch-WSPT` does not consider the complete solution space of the problem since the sequence of operations inside batches is heuristically defined. To overcome this, we propose a variation on the `Batch-WSPT` formulation by adding a new variable and constraints to define the sequence of operations inside batches without considering the subsets $\mathcal{O}_i$ (Equation 4-33), named `Batch-S`.

To control the constraints and variables generation in the model, we consider the subsets $\mathscr{M}_{i\hat{\imath}}$ of machines that are eligible and with enough capacity for executing each pair $(O_i, O_{\hat{\imath}} \in \mathcal{O})$ of operations in the same batch, defined as $\mathscr{M}_{i\hat{\imath}} = \{M_k \in (\mathcal{M}_i \cap \mathcal{M}_{\hat{\imath}}) \mid O_i \neq O_{\hat{\imath}},\ f_i = f_{\hat{\imath}},\ l_i + l_{\hat{\imath}} \leq q_k\}$. A parameter $\mu_{i\hat{\imath}} \in \{0, 1\}$ is used to identify pairs of operations with at least one machine eligible to execute both in the same batch, equals 1 if $|\mathscr{M}_{i\hat{\imath}}| > 0$, and zero otherwise. We do the same for operations triplets $(O_i, O_{\hat{\imath}}, O_{i'} \in \mathcal{O})$. Thus, $\mathscr{M}_{i\hat{\imath}i'} = \{M_k \in (\mathcal{M}_i \cap \mathcal{M}_{\hat{\imath}} \cap \mathcal{M}_{i'}) \mid O_i \neq O_{\hat{\imath}}, O_{\hat{\imath}} \neq O_{i'}, O_{i'} \neq O_i,\ f_i = f_{\hat{\imath}} = f_{i'},\ l_i + l_{\hat{\imath}} + l_{i'} \leq q_k\}$, and $\mu_{i\hat{\imath}i'} \in \{0, 1\}$, equals 1 when $|\mathscr{M}_{i\hat{\imath}i'}| > 0$, and zero otherwise.

The following binary variable $Z_{i\hat{\imath}}$ is added to sequence operations inside batches:

$$Z_{i\hat{\imath}} = \begin{cases} 1 & \text{if operation } O_i \text{ and } O_{\hat{\imath}} \text{ are scheduled in the same batch and } O_i \text{ precedes } O_{\hat{\imath}}; \\ 0 & \text{otherwise.} \end{cases}$$

The `Batch-S` formulation is as follows:

$$\min (4\text{-}34)$$

subject to

(4-35)-(4-42), (4-44)-(4-48)

$$Z_{i\hat{\imath}} + Z_{\hat{\imath}i} \geq X_{ik}^{b} + X_{\hat{\imath}k}^{b} - 1 \qquad \forall i, \hat{\imath} \in \mathcal{O}, k \in \mathscr{M}_{i\hat{\imath}}, b \in \mathcal{B}_k \quad (6\text{-}1)$$

$$Z_{i\hat{\imath}} + Z_{\hat{\imath}i} \leq 1 \qquad \forall i, \hat{\imath} \in \mathcal{O} \mid \mu_{i\hat{\imath}} \quad (6\text{-}2)$$

$$Z_{i\hat{\imath}} + Z_{\hat{\imath}i'} + Z_{i'i} \leq 2 \qquad \forall i, \hat{\imath}, i' \in \mathcal{O} \mid \mu_{i\hat{\imath}i'} \quad (6\text{-}3)$$

$$C_i \geq S_k^b + p_i + s_{f_i} + \sum_{\hat{\imath} \in \mathcal{O}} p_{\hat{\imath}} Z_{\hat{\imath}i} - \left(1 - X_{ik}^{b}\right) M \qquad \forall i \in \mathcal{O}, k \in \mathcal{M}_i, b \in \mathcal{B}_k \quad (6\text{-}4)$$

$$Z_{i\hat{\imath}} \in \{0, 1\} \qquad \forall i, \hat{\imath} \in \mathcal{O} \mid \mu_{i\hat{\imath}} \quad (6\text{-}5)$$

Constraints (6-1) identify when operations i and $\hat{\imath}$ are scheduled in the same batch. Constraints (6-2) ensure that only one of the variables that define the precedence between operations i and $\hat{\imath}$ inside a batch will be considered. Constraints (6-3) guarantee a complete ordering between operations inside a batch. Constraints (6-4) replace Constraints (4-43). The completion time of an operation is now computed with the sequencing variable $Z_{i\hat{\imath}}$. Finally, Constraints (6-5) present variable $Z_{i\hat{\imath}}$ domains.

6.2 Constructive Heuristic

We use the `WMCT-WAVGA`, presented in Chapter 5, to build the initial solutions for the matheuristics. As mentioned before, the method creates batches by assigning operations sequentially to the machines. Therefore, at each iteration, the algorithm selects the next operation to schedule and assigns a machine to the operation. Then, the method decides whether to insert the selected operation in the last batch (called current batch) or to create a new batch to insert it in, on the assigned machine. If a new batch is created, the selected operation is sequenced as the first one inside the new batch, i.e., after a new family setup time is also inserted in the machine schedule. Otherwise, the operation is scheduled as the last one in the current batch on the assigned machine. There are three situations that force the creation of a new batch on the selected machine: (1) when the machine is empty; (2) when the current batch on the machine is of a different family from the selected operation; (3) when the insertion of the operation in the current batch exceeds the machine capacity. The `WMCT-WAVGA` heuristic defines a list of schedules $\sigma = (\sigma_1, \ldots, \sigma_k)$ containing operations and families for each machine k. The families represent the setup times, indicating the beginning of a new batch.

There are no rules to control the sequence of operations inside the batches during the `WMCT-WAVGA` heuristic execution. Since the `Batch-WSPT` formulation generates these sequences heuristically, we need to update the subsets $\mathcal{O}_i$ to use the constructed solutions in this formulation. Thus, we introduce a variable $\vartheta i\hat{\imath} \in \{0,1\}$ which is equals 1 if a pair of operations $(O_i, O_{\hat{\imath}} \in \mathcal{O})$ is scheduled in the same batch and i precedes $\hat{\imath}$ in the constructed solution, and zero otherwise. Thus, the new definition of the subsets $\mathcal{O}_i$ is shown in Equation (6-6).

$$
\begin{aligned}
\mathcal{O}_i = & \Bigg\{ O_{\hat{\imath}} \in \mathcal{O} \;\Bigg|\; \vartheta\hat{\imath}i \vee \left(\vartheta i\hat{\imath} = \vartheta\hat{\imath}i \wedge \frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} > \frac{w_i}{p_i} \right) \vee \left(\vartheta i\hat{\imath} = \vartheta\hat{\imath}i \wedge \frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} = \frac{w_i}{p_i} \wedge w_{\hat{\imath}} > w_i \right) \\
& \vee \left(\vartheta i\hat{\imath} = \vartheta\hat{\imath}i \wedge \frac{w_{\hat{\imath}}}{p_{\hat{\imath}}} = \frac{w_i}{p_i} \wedge w_{\hat{\imath}} = w_i \wedge O_{\hat{\imath}} < O_i \right) \Bigg\} \quad \forall O_i \in \mathcal{O}
\end{aligned}
\tag{6-6}
$$

Suppose that the solution depicted in Figure 3.5 was generated by the `WMCT-WAVGA` heuristic. Then, the schedule σ_k for the fourth machine ($k = 4$) would be $\sigma_4 = \{f_{14}, 14, f_6, 6, f_{15}, 15, 7\}$, where f_{14}, f_6, and f_{15} represents families 1, 2, and 3, respectively. In that case, $\vartheta\hat{\imath}i = 1$ only for $O_{\hat{\imath}} = 15$ and $O_i = 7$, since this pair of operations are scheduled together in the last batch on machine M_4. In the next section, we present the MIP-based neighborhood searches to consider in the local search step of our methods.

6.3 MIP-based Neighborhood Searches

We consider two MIP-based neighborhood searches, named *Batch Windows* and *Multi-Batches Relocate*, making use of the batch formulations presented in Chapter 4, to decompose the PLSVSP into smaller problems that can be optimized more quickly than the complete problem. The idea is to limit the number of integer variables to optimize at each iteration, fixing the remaining ones from a feasible solution. As mentioned before, we consider the initial feasible solution, the one provided by the `WMCT-WAVGA` heuristic. The approaches are detailed in the next sections.

In both methods, we use variables $MB_k \in \mathbb{Z}_+$ (Machine Batches) to limit the subset of batches to be considered on each machine $M_k \in \mathcal{M}$ during the search. These variables are bounded by the size of the subsets $\mathcal{B}_k$ on each machine M_k, thus $0 < MB_k \leq |\mathcal{B}_k|$. Throughout the search, only batches with position $b \leq MB_k$ are available to optimize. The remaining batches are fixed without any operations inside them. To help to clarify the idea, we show in Figure 6.1 a graph representation with batches of the PLSV schedule illustrated in Figure 3.5 (Chapter 3). Each node represents a batch b on a machine M_k, with the respective operations assigned to it described inside. Below each node, we show the starting time (S_k^b) and the completion time (C_k^b) of the respective batch. Note that the maximum number of batches on each machine M_k is given by the total number of eligible operations. For instance, in machine M_1, we have $|\mathcal{B}_k| = 10$ (batches 1 to 10), following the subsets $\mathcal{O}_k$ defined in Table 3.4, with $\mathcal{O}_1 = \{O_1, O_2, O_4, O_5, O_7, O_9, O_{10}, O_{11}, O_{12}, O_{14}\}$. Nodes are labeled as: (1) Available batches (batches that can be used by the search procedures); (2) Unavailable batches (batches generated by the formulation but not available for the search procedure in a given iteration, according to variables MB_k). The variables MB_k are updated during the procedures, allowing an unavailable batch to become available at some point in the search.

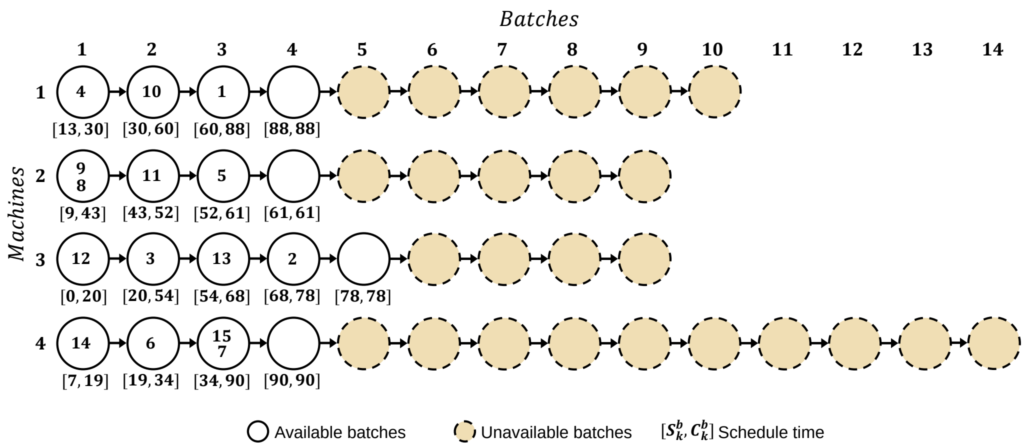


Figure 6.1: Graph representation with batches of the PLSV scheduling example.

The MB_k variables are initialized according to a solution provided by the `WMCT-WAVGA` heuristic, considering the number of used batches plus one extra batch on each machine M_k. Let $\eta_k^b \in \{0, 1\}$ be a variable equal 1 if a batch b on a machine M_k is used in a given solution (i.e., contains at least one operation scheduled in it), and zero otherwise. Then, MB_k are initialized according to Equation (6-7). Note that, on machine 1, we have four available batches (1 to 4), with $MB_1 = 4$, although the solution, depicted in Figures 3.5 and 6.1, only uses three batches on this machine. At each iteration of the MIP-based neighborhood searches, we check whether all available batches are being used on each machine M_k, updating MB_k with Equation (6-7), if $\sum_{b\in\mathcal{B}_k}\eta_k^b = MB_k$.

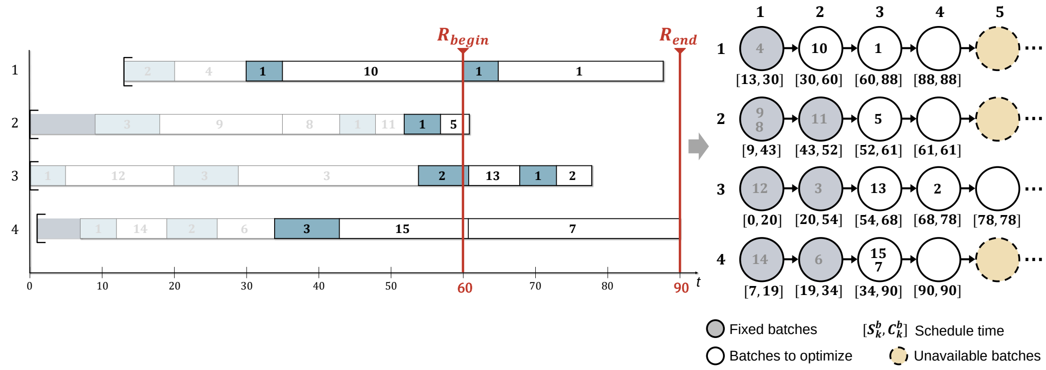
$$MB_k = 1 + \sum_{b\in\mathcal{B}_k}\eta_k^b \quad \forall M_k \in \mathcal{M} \tag{6-7}$$

6.3.1
Batch Windows

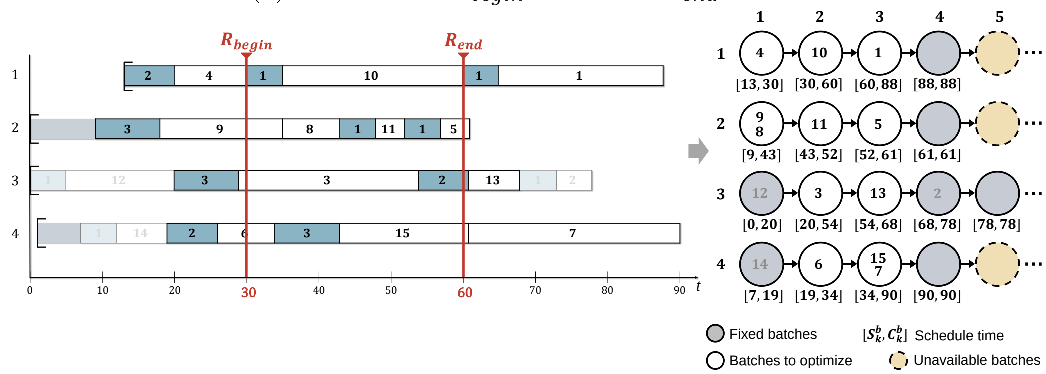
In this approach, we limit the subset of batches to optimize based on a defined time range interval. At each iteration, we only optimize batches that are scheduled inside the range. The Range Size (RS) is defined as a fraction of the makespan ($C_{max} = \max_{O_i\in\mathcal{O}} C_i$) based on a given PLSV solution, computed as $RS = \lceil \rho \times C_{max} \rceil$, where $\rho \in [0,1]$ is a parameter that defines the proportion of the makespan to consider. The complete search moves the optimization range from the end of the schedule to its beginning, with a step half the size of RS, ensuring overlap between iterations and that all batches are optimized at least once, totalizing in $\lceil C_{max}/(RS/2) \rceil$ - 1 iterations. We chose to move the search from the end of the schedule to its beginning based on preliminary experiments that showed advantages in this approach. We use R_{begin} and R_{end} to identify the beginning and the end of the optimization range to consider at each iteration.

Given the PLSV schedule example shown in Figure 3.5 (Chapter 3), we depict in Figure 6.2 how the optimization range defines the batch windows to optimize on each machine. We consider an optimization range of size 30 ($RS = 30$), which would result in a total of five iterations, with the following ranges (R_{begin}, R_{end}): Iteration 1 (60, 90); Iteration 2 (45, 75); Iteration 3 (30, 60); Iteration 4 (15, 45); Iteration 5 (0, 30). To save space, we only show iterations 1, 3, and 5, depicted in Figures 6.2a, 6.2b, and 6.2c, respectively. In this example, we suppose the solution does not change during the search. On the right side of the figures, we show the solution graph representation with the batches on each machine, highlighting the ones to optimize at each iteration. Nodes are labeled as: (1) Fixed batches (batches not selected in the given iteration); (2) Batches to optimize (batches selected to optimize in the given iteration); (3) Unavailable batches (batches generated by the formulation but not available for the search procedure in the given iteration, according to variables MB_k).

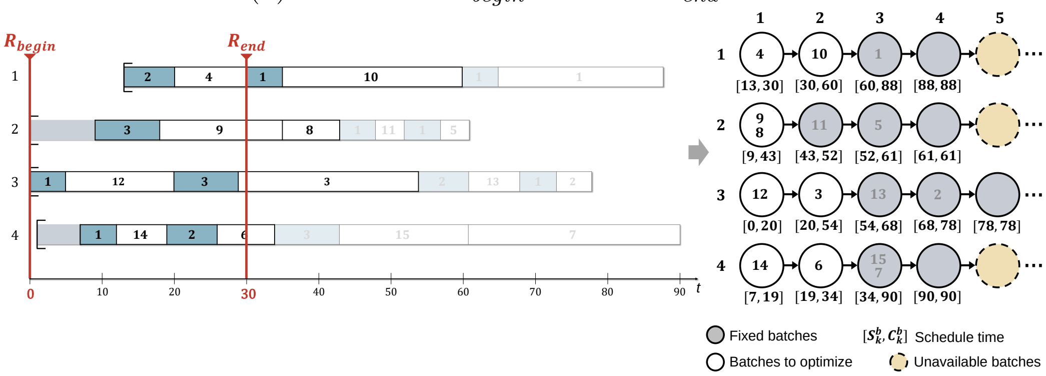
Note that a given optimization range defines different sizes of batch windows to optimize, on each machine, due to the continuous variables that



(a) Iteration 1: $R_{begin} = 60$ and $R_{end} = 90$.



(b) Iteration 3: $R_{begin} = 30$ and $R_{end} = 60$.



(c) Iteration 5: $R_{begin} = 0$ and $R_{end} = 30$.

Figure 6.2: Example of three iterations in the Batch Windows neighborhood search, showing a PLSV schedule and the optimization range on the left side, and the graph representation with the batches to optimize highlighted on the right side.

compute the starting time (S_k^b) and the completion time (C_k^b) of each available batch. For instance, in Iteration 3, depicted in Figure 6.2b, on machines 1 and 2, there are three batches (1 to 3) to optimize, while on machines 3 and 4, there are two batches (2 and 3) to optimize. Let $\mathcal{O}' \subseteq \mathcal{O}$ be the subset of operations assigned to the selected batches to optimize in a given iteration. Then, $\mathcal{O}' = \{O_1, O_3, O_4, O_5, O_6, O_7, O_8, O_9, O_{10}, O_{11}, O_{13}, O_{15}\}$ in iteration 3. The pseudo-code of the *Batch Windows* is shown in Algorithm 4.

Algorithm 4: Batch Windows ($s, \rho,\ MB_k$)

1 $C_{max} \leftarrow \max_{O_i \in \mathcal{O}} C_i$, where C_i is given by the PLSVSP solution s;
2 $RS \leftarrow \lceil \rho \times C_{max} \rceil$;
3 $R_{begin} \leftarrow \infty$;
4 $R_{end} \leftarrow C_{max}$;
5 **while** $R_{begin} > 0$ **do**
6     $C_k^b \leftarrow S_k^b + P_k^b,\ \forall M_k \in \mathcal{M}, b \in \mathcal{B}_k$;
7     $R_{begin} \leftarrow \max(0, R_{end} - RS)$;
8   Create the subset $\mathcal{O}'$ of operations assigned to batch b on machine M_k in which $S_k^b \leq R_{end}$ *and* $C_k^b \geq R_{begin}$ *and* $b \leq MB_k$ in solution s;
9   Solve *Batch Formulation*, starting from solution s, for a subset of variables X_{ik}^b in which $S_k^b \leq R_{end}$, $C_k^b \geq R_{begin}$, $b \leq MB_k$ and $O_i \in \mathcal{O}'$;
10     $R_{end} \leftarrow R_{begin} + RS/2$;
11   Update variables MB_k according to Equation 6-7, if $\sum_{b \in \mathcal{B}_k} \eta_k^b = MB_k$;
12 **end**

Algorithm 4 starts by computing values for C_{max}, RS, R_{begin}, and R_{end} (Lines 1-4), according to a given solution s and the defined parameter ρ. The main loop of the algorithm (Lines 5-12) is repeated until R_{begin} reaches zero. The completion time (C_k^b) of each batch b on each machine k is computed in Line 6. The algorithm updates C_k^b and R_{begin} (Lines 6 and 7), defining the subset $\mathcal{O}'$ in Line 8. The *Batch Formulation* is solved for all variables X_{ik}^b of batches scheduled within the optimization range (Line 9). After solving the model, the value of R_{end} is updated for the next iteration in Line 10. Finally, the procedure to update the number of available batches on each machine is executed in Line 11.

6.3.2
Multi-Batches Relocate

In this approach, we randomly select the batches to optimize at each iteration, not allowing the selection of batches already optimized in previous iterations. The complete search ends when each batch is optimized exactly once in one of the iterations. We compute the Number of Batches (NB) to optimize at each iteration based on a given parameter $\varphi \in [0, 1]$, as $NB = \lceil \varphi \times \sum_{M_k \in \mathcal{M}} MB_k \rceil$. The method runs with a total of $\lceil \sum_{M_k \in \mathcal{M}} MB_k / NB \rceil$ iterations.

An example of the *Multi-Batches Relocate* is depicted in Figure 6.3, using the graph representation shown in Figure 6.1. We consider $NB = 6$, generating a total of three iterations. At each iteration, we highlight the subset of batches to optimize. Nodes are labeled as: (1) Optimized fixed batches (batches already optimized in previous iterations); (2) Non-optimized fixed batches (batches not yet optimized but not selected in the given iteration); (3) Batches to optimize (randomly selected batches to optimize in the given iteration); (4) Unavailable batches (batches generated by the formulation but not available for the search procedure in the given iteration, according to variables MB_k).

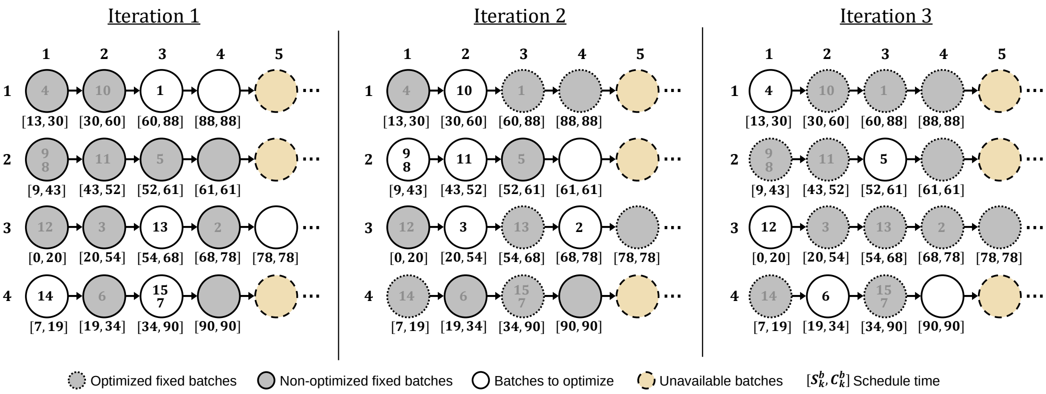


Figure 6.3: Example with three iterations of batches selection in the Multi-Batches Relocate.

Note that at the end of the search, each batch is optimized precisely once in one of the iterations. A higher diversification can be seen in this neighborhood, as it allows the selection of non-sequential batches. One can also note that in the last iteration (Iteration 3), only five batches are selected, although we defined $NB = 6$. This is because these five batches are the only ones not optimized at this point in the search. The pseudo-code of the *Multi-Batches Relocate* is shown in Algorithm 5. To describe the procedure, we use $\mathcal{P}$ to define the set of pairs machine/batch, where each element $(k, b) \in \mathcal{P}$ represents a specific batch b on a machine M_k, thus $\mathcal{P} = \{(k, b) \,|\, k \in \mathcal{M},\, b \in \mathcal{B}_k\}$.

Let $\mathcal{P}' \subseteq \mathcal{P}$ be the subset of selected pairs batch/machine on a given iteration. Algorithm 5 starts by computing the value of NB (Line 1) and initializing the set $\mathcal{P}$ (Line 2). The main loop of the algorithm (Lines 3-9) is repeated until there exist pairs batch/machine not optimized. Each iteration starts by randomly selecting NB pairs batch/machine from the set $\mathcal{P}$ to compose the subset $\mathcal{P}'$ (Line 4). The subset $\mathcal{O}'$ of operations assigned to any of the selected pairs batch/machine is created in Line 5. The *Batch Formulation* is solved, for a limited time, for all variables X_{ik}^b, where $(k, b) \in \mathcal{P}'$ and $O_i \in \mathcal{O}'$

Algorithm 5: Multi-Batches Relocate (s, φ, MB_k)

1 $NB \leftarrow \lceil \varphi \times \sum_{M_k \in \mathcal{M}} MB_k \rceil$;
2 Initialize set $\mathcal{P}$ of pairs (k, b) considering all machines $k \in \mathcal{M}$ and their respective batches $b \in \mathcal{B}_k \mid b \leq MB_k$;
3 **while** $\mathcal{P} \neq \varnothing$ **do**
4   Select NB pairs machine/batch (k, b) randomly from $\mathcal{P}$ to compose subset $\mathcal{P}'$;
5   Create the subset $\mathcal{O}'$ of operations scheduled in the subset $\mathcal{P}'$ of pairs machine/batch (k, b) on solution s;
6   Solve *Batch Formulation*, for the given solution s, for a subset of variables X_{ik}^{b} in which $(k, b) \in \mathcal{P}'$ and $O_i \in \mathcal{O}'$;
7     $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}'$;
8   Update variables MB_k according to Equation 6-7, if $\sum_{b \in \mathcal{B}_k} \eta_k^b = MB_k$;
9 **end**

(Line 6). Then, the set $\mathcal{P}$ and variables MB_k are updated in Lines 7 and 8, respectively.

6.4 Matheuristics

In this section, we present the matheuristics, which combines the *MIP-based Neighborhood Searches* (Section 6.3) and the `WMCT-WAVGA` heuristic aiming to improve solutions continuously. Two well-known algorithm frameworks from the metaheuristics literature are considered, the Iterated Local Search (ILS) and the Greedy Randomized Adaptive Search Procedure (GRASP). The following sections explain each method that we refer to as `ILS-Math` and `GRASP-Math`, respectively.

For the local search, we consider a Variable Neighborhood Descent (VND) algorithm (Hansen and Mladenović 2003), using the two *MIP-based Neighborhood Searches* described in Section 6.3. First, we run the *Multi-Batches Relocate* (Section 6.3.2), changing for the *Batch Windows* (Section 6.3.1) if no improvement is found. Every time an improved solution is found, we restart the local search, returning to the *Multi-Batches Relocate*. The local search stops when no improvement is found after running both *MIP-based Neighborhood Searches* completely. The sequence between the neighborhoods was defined based on preliminary experiments that showed a faster solution improvement using the *Multi-Batches Relocate*.

6.4.1 Iterated Local Search

ILS is a powerful tool for combinatorial optimization problems with a simple structure and very useful for practical experiments. The method starts by building an initial solution and improving it using a local search. Then, the main loop consists of perturbing the current solution with simple modifications and running the local search until a stopping criterion is reached (Lourenço et al. 2003).

Algorithm 6: ILS-Math (ρ, φ, ω, δ, Ω^{max})

Build an initial PLSVSP solution s_0 using the `WMCT-WAVGA` heuristic;
Initialize variables MB_k based on s_0 according to Equation (6-7);
$s \leftarrow VND(s_0, \rho, \varphi, MB_k)$;
$s^* \leftarrow s$;
$\Omega \leftarrow 1$;
while $\Omega \leq \Omega^{max}$ **do**
    $\Omega \leftarrow \Omega + 1$;
    $s' \leftarrow RandomBatchSwap(s, \omega, MB_k)$;
    $s'^* \leftarrow VND(s', \rho, \varphi, MB_k)$;
    if $f(s'^*) < f(s^*) \times (1 + \delta)$ **then**
        $s \leftarrow s'^*$;
        if $f(s) < f(s^*)$ **then**
            $s^* \leftarrow s'^*$;
            $\Omega \leftarrow 1$;
        end
    else
        $s \leftarrow s^*$;
    end
end
Restart variables MB_k based on the best solution s^* according to Equation (6-7);
$s^* \leftarrow VND(s^*, \rho, \varphi, MB_k)$;
return s^*;

The ILS matheuristic (`ILS-Math`) is described in Algorithm 6. A parameter $\Omega^{max} \in \mathbb{Z}_+$ defines the maximum number of iterations without improvement to execute, stopping the procedure whenever the value of the counter Ω reaches Ω^{max}. During the ILS execution, worst solutions may be accepted, according to an acceptance parameter $\delta \in [0, 1]$. We use s^* to keep the best solution found among all iterations.

The perturbation phase of the `ILS-Math` consists of randomly swapping batches in a given PLSVSP solution. In this approach, named *RandomBatchSwap*, we compute the number of swaps (NS) to be performed based on a given parameter $\omega \in [0, 1]$, where $NS = \lceil \omega \times \sum_{M_k \in \mathcal{M}} MB_k \rceil$. Note that the number of swaps to be performed is a fraction of the total number of available batches, considering all machines. At each swap movement, two batches are selected, and their operations are exchanged. If an empty batch is selected, the movement consists of removing the operations of the batch with operations inside and inserting them in the empty one. We forbid the selection of two empty batches. The movement is executed by updating the value of the corresponding assignment variables X^b_{ik} of the selected batches.

The `ILS-Math` starts by building an initial solution s_0 (Line 1), using the `WMCT-WAVGA` heuristic. In Line 2, the variable MB_k is initialized (See Section 6.4). The *VND* is then performed on s_0 (Line 3), generating the current solution s, which is copied to s^* (Line 4). The iteration counter Ω is initialized in Line 5. The main loop (Lines 6-19) is executed until $\Omega \leq \Omega^{max}$ and consists of repeatedly executing the *Random Batch Swap* (Line 8), followed by the *VND* (Line 9). In Line 10, the algorithm checks whether the objective value of the new solution s'^*, given by $f(s'^*)$, passes the acceptance criteria. If true, the current solution s is updated to s'^* (Line 11), and the algorithm checks whether this solution is also better then s^* (Line 12). If true, s^* is updated (Line 13), and Ω is reinitialized (Line 14). When the new solution is not accepted, the best solution s^* replaces s (Line 17). Finally, an intensification step is executed by running the *VND* on the best solution s^* (Lines 20-21).

6.4.2
Greedy Randomized Adaptive Search Procedure

GRASP is a multi-start method that combines a randomized constructive procedure followed by a local search, being successfully applied to many scheduling problems in the literature. For instance, we refer the reader to the papers of Bassi et al. (2012), Rodriguez et al. (2012), and Heath et al. (2013). In the constructive procedure, a Restricted Candidate List (RCL) with the most promising elements is built and one element is randomly selected at each step Resende and Ribeiro (2019). To randomize the constructive procedure for the PLSVSP, we replace the operation selection step of the `WMCT-WAVGA` heuristic by a *Randomized Operation Selection*, using a parameter $\alpha \in [0,1]$ to define the greediness of the method. When $\alpha = 0$, the method builds the same solution of the `WMCT-WAVGA` heuristic, and when $\alpha = 1$, a completely randomized solution is generated. After computing the priority value π_i for the set $\mathcal{U}$ of unscheduled operations, we identify the minimum and maximum priority values, defined as π_{min} and π_{max}, respectively. The RCL is created as RCL $= \{i \in \mathcal{U} \,|\, \pi_i \geq \pi_{max} - \alpha(\pi_{max} - \pi_{min})\}$, and one operation i^* in randomly selected from the RCL.

The pseudo-code of the GRASP matheuristic (`GRASP-Math`) is shown in Algorithm 7. We call *Randomized Constructive Procedure*, the `WMCT-WAVGA` heuristic with the *Randomized Operation Selection*. A parameter $\Omega^{max} \in \mathbb{Z}_+$ defines the maximum number of iterations without improvement, while s^* saves the best solution found among all iterations.

The `GRASP-Math` starts by initializing the Ω counter and $f(s^*)$ (Lines 1 and 2). The main loop is executed until $\Omega \leq \Omega^{max}$ (Lines 3-12). At each

Algorithm 7: GRASP-Math (ρ, φ, α, Ω^{max})

1 $\Omega \leftarrow 1$;
2 $s^* \leftarrow \emptyset$; $f(s^*) \leftarrow \infty$;
3 **while** $\Omega \leq \Omega^{max}$ **do**
4   $\Omega \leftarrow \Omega + 1$;
5   Build a PLSVSP solution s using the *Randomized Constructive Procedure*;
6   Initialize variables MB_k based on s according to Equation (6-7);
7   $s'^* \leftarrow VND(s, \rho, \varphi, MB_k)$;
8   **if** $f(s'^*) < f(s^*)$ **then**
9     $s^* \leftarrow s'^*$;
10     $\Omega \leftarrow 1$;
11   **end**
12 **end**
13 Restart variables MB_k based on the best solution s^* according to Equation (6-7);
14 $s^* \leftarrow VND(s^*, \rho, \varphi, MB_k)$;
15 **return** s^*;

iteration, a new randomized solution s is built (Line 5), with its MB_k variables initialized (Line 6), and the *VND* is applied to this solution (Line 7), generating a new solution s'^*. The algorithm checks whether the objective value of the new solution, given by $f(s'^*)$, is better than $f(s^*)$ (Line 8). If true, s^* is replaced by s'^*, and the counter Ω is reinitialized (Lines 9-10). Finally, an intensification step is executed by running the *VND* on the best solution s^* (Lines 13-14).

6.4.3 Overview of the Methodology

The general pseudocode of the proposed methodology is described in Algorithm 8. The algorithm allows selecting different mathematical formulations to be used in the matheuristics' main loop and the intensification step.

Algorithm 8: General Methodology

Choose a matheuristic framework between `GRASP-Math` and `ILS-Math`.
Select a mathematical formulation between `Batch-WSPT` and `Batch-S`.
Run the chosen matheuristic's main loop using the selected mathematical formulation.
Choose a mathematical formulation between `Batch-WSPT` and `Batch-S`.
Initialize the variables of the chosen mathematical formulation according to the best solution found so far.
Run the VND algorithm using the chosen mathematical formulation.
Return the best-found solution.

The combination of the developed matheuristics' frameworks (`ILS-Math` and `GRASP-Math`) and the batch scheduling formulations (`Batch-WSPT` and `Batch-S`), following the defined methodology, generates three variants of each matheuristic, which we refer to as $\texttt{ILS-Math}_1$, $\texttt{ILS-Math}_2$, $\texttt{ILS-Math}_3$, $\texttt{GRASP-Math}_1$, $\texttt{GRASP-Math}_2$, and $\texttt{GRASP-Math}_3$, described below:

- `ILS-Math`$_1$: `ILS-Math` with `Batch-WSPT`
- `ILS-Math`$_2$: `ILS-Math` with `Batch-S`
- `ILS-Math`$_3$: `ILS-Math` with `Batch-WSPT` and `Batch-S`
- `GRASP-Math`$_1$: `GRASP-Math` with `Batch-WSPT`
- `GRASP-Math`$_2$: `GRASP-Math` with `Batch-S`
- `GRASP-Math`$_3$: `GRASP-Math` with `Batch-WSPT` and `Batch-S`

The `ILS-Math`$_3$ and the `GRASP-Math`$_3$ consider the `Batch-WSPT` formulation for their main loop, changing for the `Batch-S` formulation at the intensification step.

6.5 Computational Experiments

In this section, we present the computational experiments conducted to assess the performance of the proposed matheuristics. We compare them with the *Mathematical Formulations* (Chapter 4) running independently. The computational experiments were performed on a machine with an Intel i7-8700K CPU of 3.70GHz and 64 GB of RAM running Linux. All methods were coded using C++ language solved by CPLEX 12.8 solver running in a single thread with MIP emphasis set to finding hidden feasible solutions. We limit the execution time to one second per CPLEX call in the matheuristics. Thus, sub-problem optimizations are interrupted within the defined time-limit even if the optimal solutions have not been reached. The experiments were conducted on the benchmark of 72 PLSVSP instances described in Chapter 4.

To evaluate the solution's quality, we compare the solutions provided by each method with the Best-Know Solutions (BKS), achieved by the mathematical formulations (Chapter 4), in terms of the Relative Percentage Deviation (RPD), computed according to Equation (7-1). $TWCT^{Method}$ designates the total weighted completion time obtained with one run of a selected method on a PLSVSP instance, while $TWCT^{BKS}$ is the total weighted completion time for the Best-Know Solution for the same instance. In all analyzes, we compare the matheuristics, running each one ten times per instance, with the solutions of `Batch-WSPT` and `Batch-S` formulations, running within a 6-hour time-limit. We limit the memory allocation to 10 GB for each method execution to allow multiple runs simultaneously using the available CPUs.

$$RPD = 100 \times \frac{TWCT^{Method} - TWCT^{BKS}}{TWCT^{BKS}} \tag{6-8}$$

The discussion of the results focuses only on comparing the `ILS-Math`$_3$ and the `GRASP-Math`$_3$ matheuristics with the pure mathematical formulations since these approaches presented the best results among the proposed matheuristic variants in a preliminary analysis, shown in Appendix D.1. We also included the complete results for each instance, considering all methods, in Appendix D.2.

6.5.1 Parameter Tuning

For parameter tuning, we used 12 medium-sized instances (25 operations) and 12 large-sized instances (50 operations), selected at random, corresponding to one-third of the total number of instances available in the benchmark. The `Batch-WSPT` formulation was used within the matheuristics during the parameterization. First, we set parameters related to the *MIP-based Neighborhood Searches* (Section 6.3), to further define each specific matheuristic parameter. To establish the neighborhoods' parameters, we ran five times each neighborhood individually, using the solution generated by the `WMCT-WAVGA` heuristic as a warm start. We define the ranges for the *Batch Windows* (Section 6.3.1), and for the *Multi-Batches Relocate* (Section 6.3.2) parameters as $\rho \in [0.1, 0.5]$, and $\varphi \in [0.1, 0.5]$, respectively, considering a step of size 0.05. After setting values for ρ, and φ, the `ILS-Math` was executed five times without accepting worse solutions. We used $\Omega^{max} = 10$, limiting $\omega \in [0.05, 0.15]$ with a step size of 0.05, to define the perturbation parameter. With ω set, we ran the `ILS-Math` again five more times, limiting $\delta \in [0.00, 0.15]$ with a step size of 0.05, to define the worst solution acceptance rate. The same steps were executed to define the greediness factor of the `GRASP-Math`, limiting $\alpha \in [0.05, 0.15]$, with a step size of 0.05. The final values for the parameters are shown in Table 7.1.

6.5.2 Results Analysis and Discussion

6.5.2.1 Average Values for the RPD and Computational Time

In the first analysis, shown in Table 6.2, we compare the four methods (`Batch-WSPT`, `Batch-S`, `ILS-Math`$_3$ and `GRASP-Math`$_3$) in terms of the average RPD ($\overline{RPD}$) and the average computational time ($\overline{time}$), in seconds, for each instance group. We use the same grouping scheme defined in Chapter 4 for this

Table 6.1: Parameters definition

Algorithm	Parameter	Description	Domain	Value
Batch Windows	ρ	The proportion of the makespan (C_{max}) to optimize at each iteration	[0, 1]	0.20
Multi-Batches Relocate	φ	The proportion of batches to optimize at each iteration	[0, 1]	0.30
Random Batch Swap	ω	The proportion of batch swaps to execute at each perturbation	[0, 1]	0.10
ILS-Math	δ	Worse solutions acceptance rate	[0, 1]	0.00
GRASP-Math	α	Greediness factor for the randomized constructive procedure	[0, 1]	0.10
ILS-Math and GRASP-Math	Ω^{max}	Maximum number of iterations without improvement	$\mathbb{Z}_+$	10

analysis, in which each group, represented by the combination of the number of operations and machines (o and m), comprises 12 instances. The best result for each criterion in each group is highlighted in bold.

Note that the average relative percentage deviations are low for instances with 15 operations, with an $\overline{RPD}$ below 1% for all methods. Regarding group 15–4, the `Batch-S` formulation presented the smallest $\overline{RPD}$ with 0.00, that is, it reaches the BKS in all runs, but with an average computational time of 20,725 seconds. The average computational times for the MIP formulations in this group are close to the limit of 21,600 seconds (6 hours) previously defined. $\texttt{ILS-Math}_3$ runs with the least average computational time in this group (14 seconds). However, concerning the $\overline{RPD}$, $\texttt{GRASP-Math}_3$ is the best matheuristic approach ($\overline{RPD} = 0.30$), without significantly increasing computational time spent by $\texttt{ILS-Math}_3$. Similar behavior can be seen in group 15–8. $\texttt{GRASP-Math}_3$ presented equivalent results with `Batch-S` ($\overline{RPD} = 0.01$), but consuming less than a minute to achieve it. In contrast, `Batch-S` run for an average of $18,000$ seconds.

For medium-sized groups (25–4 and 25–8), the matheuristics outperform the MIP formulations in terms of solution quality, with considerably less computing time. The MIP formulations maintained a good quality of the solutions, below 1% for the $\overline{RPD}$ in both groups, but running until the time limit in all executions ($\overline{time} = 21{,}600$). Between the matheuristics, $\texttt{GRASP-Math}_3$ outperforms $\texttt{ILS-Math}_3$ in terms of solution quality, with $\overline{RPD} = -0.31$ in group 25–4, and $\overline{RPD} = -0.27$ in group 25–8, but with 51% extra computational time needed on average in the former group (186 seconds against 123 seconds), and

Table 6.2: Average values for the RPD and computational time distributions for each method in each instance group.

o	m	Batch-WSPT		Batch-S		ILS-Math$_3$		GRASP-Math$_3$	
		$\overline{RPD}$	$\overline{time}$	$\overline{RPD}$	$\overline{time}$	$\overline{RPD}$	$\overline{time}$	$\overline{RPD}$	$\overline{time}$
15	4	0.49	20414	**0.00**	20725	0.42	14	0.30	26
	8	0.04	18000	**0.01**	18000	0.08	46	**0.01**	54
25	4	0.70	21600	0.81	21600	-0.17	**123**	**-0.31**	186
	8	0.23	21600	0.62	21600	-0.20	**244**	**-0.27**	297
50	4	-0.27	19978†	0.60	21600	**-3.02**	**1052**	-2.65	1159
	8	-1.83	17380†	1.90	13901†	**-4.09**	**990**	-3.80	1066
All instances		-0.11	19829†	0.66	19571†	**-1.16**	**412**	-1.12	465

† CPLEX execution interrupted before 21,600 seconds (time limit) for some instances in this group, for this method, due to memory problems.

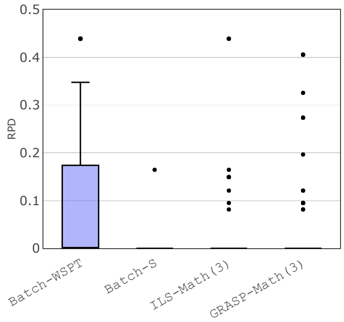
22% in the latter (297 seconds against 244 seconds).

For large-sized groups, we can see that Batch-S loses performance, with the worst values for the $\overline{RPD}$ in both groups (0.60 in group 50–4 and 1.90 in group 50–8), resulting in a clear advantage for Batch-WSPT. Again, the matheuristics dominates the MIP formulations in both groups with lower values for the $\overline{RPD}$ and $\overline{time}$. However, in these groups, ILS-Math$_3$ dominates GRASP-Math$_3$, in terms of solution quality and computational time. It can be noted that the $\overline{time}$ for the MIP formulations on these groups is smaller than the limit of 21,600. This is due to memory issues, with the solver interrupting the execution of 6 instances, in each formulation, before reaching the time limit. Appendix D.2 details the complete results, indicating these instances.

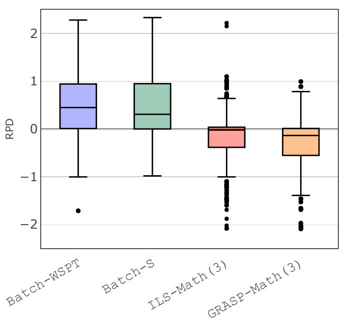
One can note that, in all groups, the GRASP approach requires more computational time than ILS due to the method's restart feature. This characteristic also affects the quality of the method's solutions when the size of the instances increases. We can also observe that the average computational time for the matheuristics grows considerably when the number of operations to schedule increases. It is known that mathematical models are severely affected when the size of problems grows, which, in our case, directly impacts the execution time of our matheuristics.

6.5.2.2
RPD Distribution for Different Instances Aspects

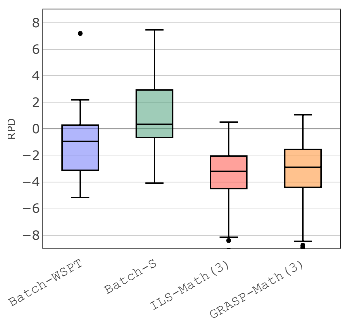
In this analysis, we evaluate the impacts on the solution quality of each method with the increasing number of machines and operations. Figures 6.4a, 6.4b, and 6.4c depict the distribution of the RPDs for instances with 15, 25, and 50 operations to schedule, respectively. Next, Figures 6.5a, and 6.5b, show the distribution of the RPDs for instances with 4, and 8 machines, respectively.



(a) 15 operations.



(b) 25 operations.

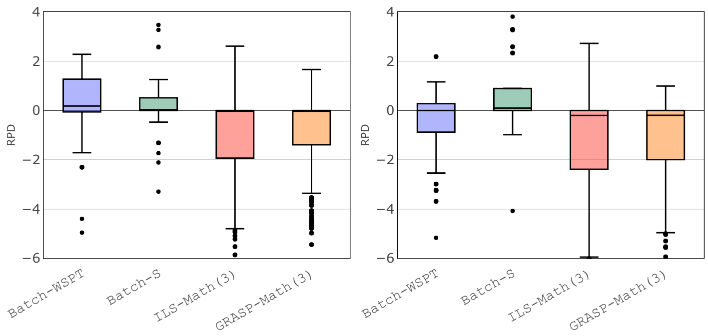


(c) 50 operations.

Figure 6.4: Boxplots of the RPD distributions for each method, considering different number of operations.

Regarding instances with 15 operations (Figure 6.4a), all methods are competitive (RPDs closest to zero). However, `Batch-WSPT` struggles to achieve the BKS due to the reduced solution space considered by the formulation. Note that the matheuristics have better performance than the MIP formulations when 25 and 50 operations are considered (Figures 6.4b and 6.4c). It is interesting to see the distributions' behavior when the number of operations increases. For the MIP formulations, we can see that `Batch-S` performs better on small instances, while `Batch-WSPT` shows advantages when the number of operations increases. Note that on medium-sized instances (25 operations), they have similar performance. The consideration of extra variables and constraints to sequence operations within batches proved to be a good strategy for small instances since it considers the complete solution space of the problem, however affecting the performance of `Batch-S` when the number of operations increases. Regarding the matheuristics, we can see that they maintain similar distributions regardless of the number of operations, but with $\texttt{ILS-Math}_3$ improving its performance, compared to $\texttt{GRASP-Math}_3$ in instances with 50 operations.

The same behavior can be observed in the analysis by the number of machines (Figures 6.5a and 6.5b), with the matheuristics showing better distribution of RPDs. However, we can observe less dispersed distributions due to the consideration of instances with 15 operations in both groups. The MIP formulations maintained the same behavior, with Batch-S losing performance when the number of machines grows.



(a) 4 machines. (b) 8 machines.

Figure 6.5: Boxplots of the RPD distributions for each method, considering different number of machines.

6.5.2.3
Comparison with the BKS Values

Now, we analyze the methods, in Table 6.3, regarding the achievement or improvement of the Best-Know Solutions from the literature, defined by the mathematical formulations in Chapter 4. The results indicate the number of instances in which the BKS was achieved or improved (#Inst.), its percentage regarding the complete set of 72 PLSVSP instances (%Inst.), and the respective percentage of runs of achievement or improvement (%Runs). The $\overline{RPD}$ for the subsets of not achieved and not improved solutions are included. Finally, we show the minimum RPD (RPD^{-}) found by each method when the BKS is improved.

Table 6.3: Analysis regarding BKS solutions from the literature.

Method	Achieved BKS[†]			Not Achieved	Improved BKS				Not Improved
	#Inst.	%Inst.	%Runs	$\overline{RPD}$	#Inst.	%Inst.	%Runs	RPD^{-}	$\overline{RPD}$
`Batch-WSPT`	42	58.33	58.33	1.16	23	31.94	31.94	-5.17	0.71
`Batch-S`	41	56.94	56.94	2.06	13	18.06	18.06	-4.08	1.08
ILS-Math$_3$	69	95.83	81.94	0.72	40	55.56	49.72	-10.96	0.26
GRASP-Math$_3$	72	100.00	86.11	0.59	40	55.56	50.56	-9.72	0.17

[†] This subset includes solutions that have also improved the BKS.

Note that all methods achieve the BKS in more than 50% of instances, with GRASP-Math$_3$ reaching it in all instances (#Inst. = 72). The %Inst. and %Run has the same value for each MIP formulation, as we only run it once for each instance. Again, the matheuristics have better performance than the MIP formulations. Note that the $\overline{RPD}$ for runs where BKS was not found is less than 1% for the matheuristics and more than 1% for the MIP formulations, with the worst performance for `Batch-S` formulation ($\overline{RPD} = 2.06\%$). It can be seen that the matheuristics ILS-Math$_3$ and GRASP-Math$_3$ achieve the BKS in

81.94% and 86.11% of the executions, respectively. The same advantage for the matheuristics can be seen for improved solutions, with both methods improving the BKS in about 50% of runs. The $\overline{RPD}$ is low for runs that have not improved the BKS (0.26% for `ILS-Math`$_3$, and 0.17% for `GRASP-Math`$_3$), as it includes runs that have achieved the BKS. With these results, we can see that 40 new best solutions are found by the matheuristics, with an improvement of 10.96% in the objective value of the solution in the best case (RPD^- of `ILS-Math`$_3$). The specific instances with enhanced solutions are indicated in Appendix D.2. Considering that 26 instances have the proven optimal solutions (see Table D.3), 40 new best solutions have been defined in 46 possible instances.

6.5.2.4
Statistical Analysis for the RPD Distributions

To improve our discussion and validate what we highlighted during the previous analyses, we applied a statistical evaluation comparing the RPD distributions between the methods, considering the complete set of 72 PLSVSP instances. First, we tested the distributions' normality with the Shapiro-Wilk test, which shows that the RPDs do not follow a normal distribution. Then, we ran the pairwise Wilcoxon rank-sum test with Hommel's p-values adjustment. We also included the Analysis of Variance (ANOVA) with the Tukey HSD (honestly significant difference) test to compare the method's RPD distributions. We ran both tests with a confidence level of 0.05. In Table 6.4, we present the p-value for each pair of methods, also including some statistics on the RPD distributions. Significantly better results are highlighted in bold.

Table 6.4: The mean and standard deviation of the RPD distributions for each method, and p-values from pairwise Wilcoxon rank-sum and ANOVA with Tukey HSD tests, with a 0.05 confidence level, considering all 72 PLSVSP instances.

Method	$\overline{RPD}$	σ	Wilcoxon p-value (ANOVA–Tukey p-value)		
			`Batch-WSPT`	`Batch-S`	`GRASP-Math`$_3$
`Batch-WSPT`	-0.10	1.78			
`Batch-S`	0.66	2.10	0.38 (0.14)		
`GRASP-Math`$_3$	-1.16	2.18	**0.00 (0.00)**	**0.00 (0.00)**	
`ILS-Math`$_3$	-1.12	2.15	**0.00 (0.00)**	**0.00 (0.00)**	0.50 (0.98)

The tests confirm that the matheuristics yield significantly better results than the MIP formulations. Note that no statistical significance can be seen when comparing one MIP formulation with another, nor when comparing the matheuristics between each other. Wilcoxon and ANOVA–Tukey agreed in all cases despite discrepancies in the p-values.

6.5.2.5
Average RPD Evolution Analysis

As we evaluate solutions for a realistic process, it is important to analyze the final solutions of the proposed methods and their evolution over time, in case the company needs faster solutions. Based on this, we show in Figure 6.6 the evolution of the solution for the two best approaches, `ILS-Math`$_3$ and `GRASP-Math`$_3$, illustrating the trade-off between the quality of the solution and the time spent for achieving it. For better visualization, the computational time axis is on a $\log_{10}$ scale. As the deviations and computational times are low for small-sized instances, the curve's shape is more defined by the medium and large-sized instances.

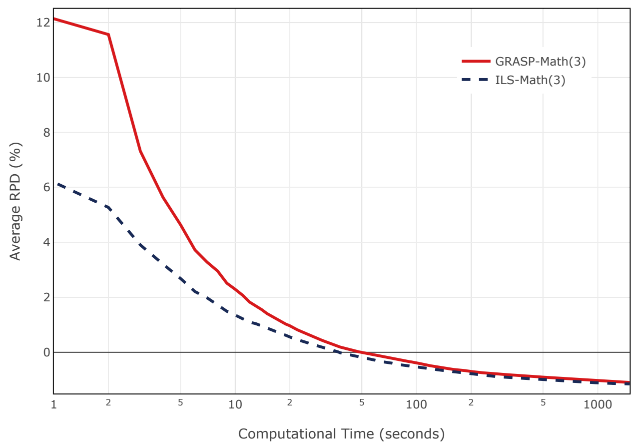


Figure 6.6: Average RPD (vertical axis) evolution over the computational time (horizontal axis – $\log_{10}$ scale) for each matheuristic, running on the complete set of 72 PLSVSP instances.

Note that both matheuristics surpass the BKS on average (that is, the current method used to solve the PLSVSP), crossing the zero line, in less than 1 minute (38 seconds for `ILS-Math`$_3$, and 49 seconds for `GRASP-Math`$_3$). The `ILS-Math`$_3$ matheuristic dominates the `GRASP-Math`$_3$ but reaching an equivalent average RPD after crossing the zero line. This analysis reinforces the advantages of using the matheuristics concept instead of running the MIP formulations, showing that the additional time-limit criteria can be included, if necessary, without significantly impacting the quality of the solution.

6.6
Discussion

In this chapter, we introduced an ILS and a GRASP matheuristics, using two MIP-based neighborhood searches and a constructive heuristic to solve the PLSVSP. Two MIP formulations are considered, resulting in three variants of each method. The first is a batch formulation, which uses a WSPT dispatching rule to sequence operations within batches, while the second is a new formulation that considers sequencing within batches as a model decision. The results show that the matheuristics outperform the pure mathematical programming models in terms of computational time and solution quality. Among the matheuristics, a small advantage can be observed for two variants that combine the use of two batch formulations for the PLSVSP. The analysis shows that our new proposed formulation with sequencing variables helps the matheuristics to improve the quality of the solution. Moreover, new best solutions are provided for 40 of the 46 possible instances (without proven optimal solutions) on the benchmark set of 72 PLSVSP instances.

The present study reinforces the importance of hybrid methods and their applicability in practical and theoretical contexts. The concept of splitting the problem to solve sub-problems is a relevant approach, especially when the size of the problem increases. It is worth mentioning that the approach is closely related to the nature of the PLSVSP in the studied company. In this environment, management guidelines for dealing with the problem change rapidly due to political and operational issues, requiring a quick adjustment in decision support tools. The use of mathematical formulation facilitates these adjustments, allowing the inclusion or exclusion of constraints without the need for substantial computational development in the algorithms.

7
Iterated Greedy Algorithm for the PLSVSP

In this chapter, we present an Iterated Greedy (IG) algorithm to solve the PLSVSP, which extends the Iterated Local Search (ILS) metaheuristic introduced by Mecler (2020). The IG approach (Ruiz and Stützle 2007) combines an intensification step, given by a local search procedure to achieve local optimal solutions, with destroying and repairing phases to diversify solutions and avoid the method being stuck. One of the main differences from the metaheuristic of Mecler (2020) is to replace the ILS perturbation step with the destroy and repair steps. Another difference is that we avoid restarting the algorithm to improve a single solution generated by the best constructive heuristic known for the PLSVSP. The remaining diversification and intensification elements of the method are the ones proposed by Mecler (2020) as the Random Variable Neighborhood Descent (RVND) local search phase, the simulated annealing acceptance criterion, the infeasibility strategy, and the restore step. However, we performed new experiments to set the algorithm's parameters.

7.1
IG algorithm pseudo-code and description

The pseudo-code of the IG approach is shown in Algorithm 9. The algorithm uses $\mathtt{s} = (\mathtt{s}_1, \ldots, \mathtt{s}_m)$ to identify a solution composed by a list of schedules for the m machines. Each schedule is a permutation of families, which indicates a setup time, and operations. The algorithm considers $\mathtt{s}$ as the iteration current solution, and $\mathtt{s}^*$ as the best overall solution during the execution. And, function $f(.)$ returns the total weighted completion time of a given PLSVSP solution. Figure 7.1 depicts an example with five operations (O_1 to O_5) and two machines (M_1 and M_2) of the solution representation considered within the algorithm. As mentioned before, the family index represents a setup time, indicating a new batch. To contextualize the example, let's consider the following information:

1. All machines are eligible for all operations, i.e. $\mathcal{M}_i = \mathcal{M}$, $\forall i \in \mathcal{O}$.

2. Three jobs are considered (J_1, J_2, and J_3), with the following weights: $w_1 = 3$, $w_2 = 2$, and $w_3 = 1$.

3. The jobs associated with each operation are: $\mathcal{N}_1 = \{J_1\}$, $\mathcal{N}_2 = \{J_2, J_3\}$, $\mathcal{N}_3 = \{J_3\}$, $\mathcal{N}_4 = \{J_1, J_2\}$, and $\mathcal{N}_5 = \{J_2\}$.

4. Operation's families are: $f_1 = f_2 = f_3 = f_4 = 1$ and $f_5 = 2$.

5. Operation's processing times are: $p_1 = p_2 = p_3 = 15$ and $p_4 = p_5 = 20$.

6. The size of the operations are: $l_1 = 30$, $l_2 = 40$, $l_3 = 10$, and $l_4 = l_5 = 40$.

7. Release dates of operations are: $r_1 = 5$, $r_2 = 0$, $r_3 = 30$, $r_4 = 0$, $r_5 = 40$.

8. Release dates of machines are: $r_1 = 0$ and $r_2 = 5$.

9. The capacity of the machines are: $q_1 = 80$ and $q_2 = 90$.

10. Family setup times are: $s_1 = 10$ and $s_2 = 5$.

In the example, each machine executes two batches, respecting their capacities. From the defined permutation and considering the described data, the Total Weighted Completion Time (TWCT) of the example, following the jobs index order (J_1, J_2, J_3), is given by: TWCT = (3 × 45) + (2 × 65) + (1 × 70) = 335.

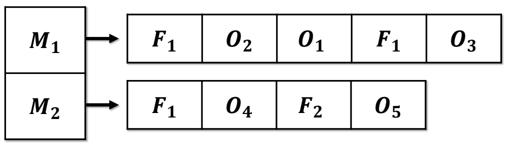


Figure 7.1: Solution representation used by the IG algorithm

First, Algorithm 9 computes the value of Ω, which indicates the maximum number of consecutive iterations without improvement allowed until the algorithm restores the current solution to the best solution found (Line 1). Then, it builds a solution `s` using the `WMCT-WAVGA` constructive heuristic (Line 2). After running the local search step (Line 3), the best overall solution and the counter ω of consecutive iterations without improvement are initialized (Lines 4–5). The main loop (Lines 6–23) runs for η iterations following three main steps: (1) Destroy and repair current solution `s`, generating solution `s'` (Lines 7–8); (2) Run the local search in `s'` to generate the local optimum solution (Line 9), checking its feasibility (Line 11). (3) Update the current solution `s` if the objective value of `s'` is better than the objective value of `s` or if the acceptance criterion is met (Lines 12–13). If the new current solution `s` is feasible and its objective value is better than the objective value of the best overall solution $\mathtt{s}^*$, $\mathtt{s}^*$ is replaced by $\mathtt{s}$ and the counter ω of consecutive iterations without improvement is reinitialized (Lines 14–17). The algorithm employs a restore strategy after ω consecutive iterations without improvement (Lines 19–22). It returns the best overall solution found $\mathtt{s}^*$ (Line 24).

Algorithm 9: Iterated Greedy Algorithm

Input : Number of iterations (η); Restore parameter (λ).
Output: The best solution found $\mathtt{s}^*$ as a list of schedules for each machine.

```
1  Ω ← ⌈λη⌉;
2  s ← Constructive();                          ▷ Construct the initial solution
3  s ← RVND(s);                                               ▷ Local Search
4  s* ← s;                                 ▷ Initialize the best overall solution
5  ω ← 0;
6  for η iterations do
7      s′ ← Destroy(s);
8      s′ ← Repair(s′);
9      s′ ← RVND(s′);
10     ω ← ω + 1;
11     is_feasible ← Feasible(s′);                     ▷ Infeasibility strategy
12     if f(s′) < f(s) or Accept(s′, s) then             ▷ Acceptance Criterion
13         s ← s′;
14         if f(s) < f(s*) and is_feasible then             ▷ Improvement check
15             s* ← s;
16             ω ← 0;
17         end
18     end
19     if ω = Ω then                                      ▷ Solution restore
20         s ← s*;
21         ω ← 0;
22     end
23 end
24 return s*;
```

The next sections detail the local search and its neighborhoods, the destroy and repair operators, the simulated annealing acceptance criterion, and the infeasibility strategy.

7.1.1 RVND Local Search

The RVND strategy, introduced by Subramanian et al. (2010), picks neighborhoods up randomly from a pool instead of running them in a deterministic pre-defined sequence as the original VND. The `RVND` local search (Lines 3 and 9 of Algorithm 9) returns a local optimum solution after running four neighborhoods, described as follows (Mecler et al. 2021):

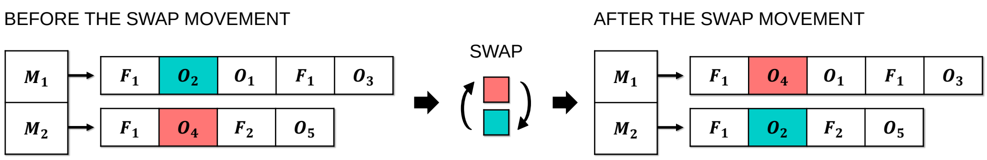
1. `Swap`: Exchange any two operations in the schedule, assigned to the same or different machines, respecting the eligibility constraint. Figure 7.2

exemplifies a swap movement between operations from the same family (Figure 7.2a), and between operations from distinct families (Figure 7.2b). The procedure includes new setup times whenever a movement generates batches with mixed families and removes extra setup times when needed to respect the family constraint.

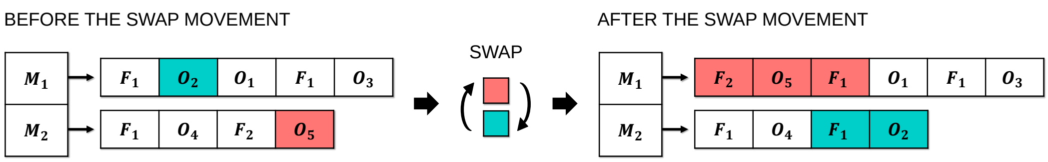
2. `Relocate`: Remove an operation assigned to any machine and reinsert it in a new position on the same or another machine, respecting the eligibility constraint. Figure 7.3 depicts a relocate movement of an operation in a batch of the same family (Figure 7.3a), and the relocation of an operation in a batch of a distinct family (Figure 7.3b). Again, the procedure includes new setup times when needed to respect the family constraint, removing extra setup times when a movement creates a solution with two consecutive setup times.

3. `SplitBatches`: Split a batch into two batches of the same family by inserting a setup time between two consecutive operations of the same family. Figure 7.4a exemplifies the movement.

4. `MergeBatches`: Merge two consecutive batches of the same family by removing the setup time of the second batch. Figure 7.4b exemplifies the movement.



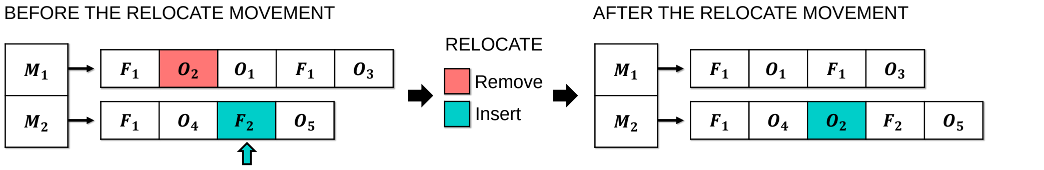
(a) Swap between operations of the same family.



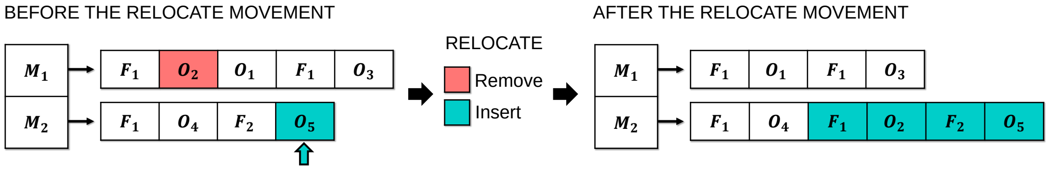
(b) Swap between operations of distinct families.

Figure 7.2: Swap movement examples (Mecler et al. 2021).

The `RVND` procedure (Algorithm 10) starts by initializing the set $\mathcal{L}$ with the indices of the neighborhoods (Line 1) as $\mathcal{L} = \{1, 2, 3, 4\}$, considering the four neighborhoods previously described (1–`Swap`, 2–`Relocate`, 3–`SplitBatches`, 4–`MergeBatches`). The main loop is executed until the set $\mathcal{L}$ is empty (Lines 2–17). The algorithm picks a neighborhood index ℓ at random (Line 3) and tests each solution in the neighborhood N_ℓ following a first

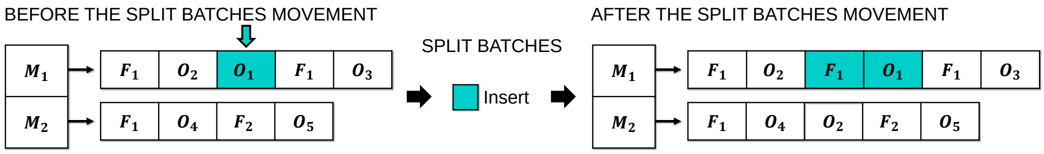


(a) Relocate in a batch of the operation's family.

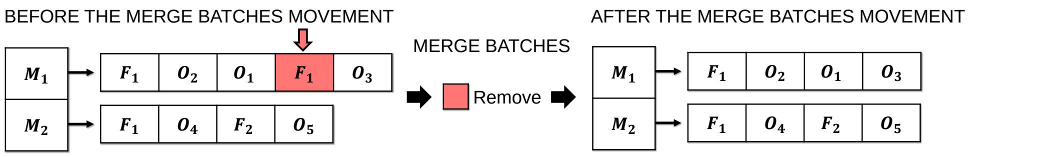


(b) Relocate in a batch of a distinct families.

Figure 7.3: Relocate movement examples (Mecler et al. 2021).



(a) Split Batches movement.



(b) Merge Batches movement.

Figure 7.4: Setup movement examples (Mecler et al. 2021).

improvement rule to update the current solutions and resets the set $\mathcal{L}$ of neighborhood indices. (Lines 5–11). If no improvement is found, the algorithm removes the current neighborhood index from set $\mathcal{L}$ (Lines 12–16). The algorithm returns the local optimal solution (Line 18).

7.1.2 Destroy and Repair Operators

The `Destroy` procedure (Line 7 of Algorithm 9) removes $d = \lceil \varepsilon o \rceil$ operations from a given solution `s` and generates a partial solution with $o - d$ scheduled operations, where $\varepsilon \in [0, 1]$ is the destruction parameter. The procedure removes extra setup times when necessary. The objective function value of a partial solution only considers the scheduled operations. Thus, we define the completion time of all unscheduled operations' as zero. The procedure anticipates the starting time of batches after removing the operations, if possible. Removed operations compose a list of unscheduled operations, following the order in which the destroy operator extracted them

Algorithm 10: `RVND` (Mecler et al. 2021)

Input : A given solution `s`.
Output: A local optimal solution `s`.

```
Initialize set ℒ of neighborhood indices;
while ℒ ≠ ∅ do
    ℓ ←random(ℒ);
    improved ← false;
    for s′ ∈ N_ℓ(s) do
        if f(s′) < f(s) then
            s ← s′;
            improved ← true;
            break;
        end
    end
    if improved then
        Initialize set ℒ of neighborhood indices;
    else
        ℒ ← ℒ \ {ℓ};
    end
end
return s;
```


from `s`. According to their sequence within the list of unscheduled operations, the repair operator picks the unscheduled operations, one-by-one to reinsert them in the solution, generating a new complete solution. We consider the following destroy and repair operators (Ruiz and Stützle 2007):

1. `RandomDestroy`: Removes operations randomly.
2. `GreedyRepair`: Reinserts each operation in the best position among all eligible machines according to the objective function value.

Figure 7.5 exemplifies the `RandomDestroy` and `GreedyRepair` operators, considering the example depicted in Figure 7.1, with $d = 2$. Note that the destruction step removes operations O_5 and O_2 from the solution, generating a partial schedule with a TWCT of 230. Then, the repair step reinserts operation O_5 in its best position, which, in this case, is its original position. A setup time is included before operation O_5. Finally, the repair operator inserts operation O_2 in the first batch on machine M_1 before operation O_1, generating a new complete solution with a TWCT of 305, better than the initial solution.

To help the reader understanding the example given, we describe the computing of the TWCT regarding the destroyed solution. We use C_i^O to indicate the completion time of operation O_i and C_j^J for the completion time of job J_j, to avoid ambiguity. We also use r_i^O to identify the release date of operation O_i and r_k^M for the release date of machine M_k. As mentioned previously, the destroyed solution leads to a TWCT of 230. The TWCT considers the completion times of the scheduled operations: $C_1^O = 30$ ($r_1^O + s_1 + p_1$), $C_3^O = 55$ ($C_1^O + s_1 + p_3$), and $C_4^O = 35$ ($r_2^M + s_1 + p_4$). As highlighted before, the completion times of unscheduled operations are defined as zero ($C_2^O = C_5^O = 0$). Considering the set of jobs associated with each operation, the completion times of jobs is then computed as: $C_1^J = \max\{C_1^O, c_4^O\} = 35$, $C_2^J = \max\{C_2^O, C_4^O, C_5^O\} = 35$, and $C_3^J = \max\{C_2^O, C_3^O\} = 55$. The TWCT is then computed as: $w_1 \times C_1^J + w_2 \times C_2^J + w_3 \times C_3^J = 3 \times 35 + 2 \times 35 + 1 \times 55 = 230$.

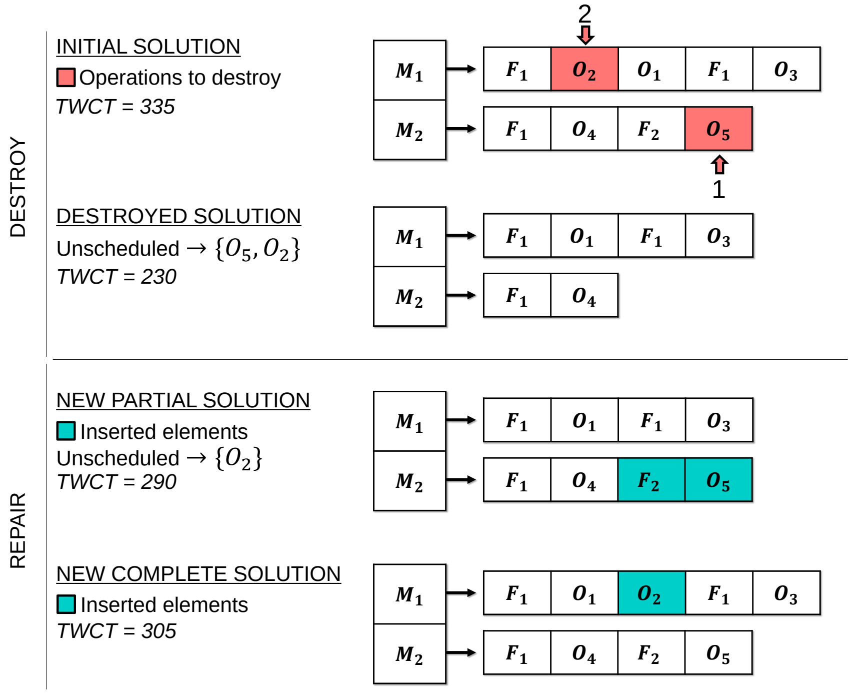


Figure 7.5: Destroy and repair example with $d = 2$ (Mecler et al. 2021).

7.1.3 Acceptance Criterion

The `Accept` procedure (Line 12 of Algorithm 9) employees a simulated annealing criterion to decide whether a candidate solution $\mathtt{s}'$ should replace solution $\mathtt{s}$ as the current solution, accepting $\mathtt{s}'$ with probability $e^{-\Delta/\tau}$, where $\Delta = f(\mathtt{s}') - f(\mathtt{s})$, and τ is the current temperature. The temperature τ is initialized with a value τ_0 and decreases at each iteration as $\tau = \tau\kappa$, where $\kappa = [0, 1)$ is the cooling rate (Kirkpatrick et al. 1983). The initial and final temperatures (τ_0 and τ_F) are instance-dependent, as proposed by Pisinger and Røpke (2007), computed as $\tau_0 = -(\delta_1 f_o)/\ln(0.5)$ and $\tau_F = -(\delta_2 f_o)/\ln(0.5)$, respectively, where f_0 is the objective value of the initial solution, and δ_1 and δ_2 are adjustable parameters. The initial and final temperature are defined to accept, with 50% probability, solutions δ_1 and δ_2 worse than the initial solution, respectively. The cooling rate κ is set by considering the number of iterations η to execute and the initial and final temperatures, computed as $\kappa = (\tau_F/\tau_0)^{1/\eta}$.

7.1.4 Infeasibility Strategy

The algorithm uses the infeasibility strategy of Mecler (2020) to enlarge the problem's search space, increase diversification, and escape from local optimal solutions. In this strategy, infeasible solutions regarding the capacity might be accepted. The cost of a solution $\mathtt{s}$ with capacity violations is updated as $f(\mathtt{s}) = f(\mathtt{s}) + \rho V$, where $\rho \geq 1$ is the penalty factor, V is the total capacity violation among all batches from all machines, and $f(\mathtt{s})$ is the objective function value of solution $\mathtt{s}$. Parameters $\rho^+ \in [0, 1)$ and $\rho^- \in [0, 1)$ are used to update the value of ρ at each iteration. Thus, when the algorithm accepts a new current solution, the penalty factor is updated as $\rho = \rho(1 + \rho^+)$, if the new current solution is infeasible, and as $\rho = \rho(1 - \rho^-)$, if feasible. Thus, the algorithm increases the penalty factor whenever it accepts infeasible solutions to prioritize feasible solutions. The `Feasible` procedure (Line 11 of Algorithm 9) updates the parameter ρ after verifying if a given solution is feasible, returning `true` when no capacity violations exists, and `false`, otherwise.

7.2 Computational Experiments

In this section, we conduct computational experiments on the benchmark set of 72 PLSVSP instances to evaluate the performance of the proposed IG algorithm, named `IG-RG`. The algorithm's input parameters are calibrated, and the method is compared with the matheuristics presented in Chapter 6. We use C++ language for coding the IG algorithm, and ten independent runs are performed within the experiments. All experiments are performed on a computer with an Intel i7-8700K CPU of 3.70GHz and 64 GB of RAM, running Linux with a single thread.

In all analyses, we evaluate solutions in terms of the Relative Percentage Deviation (RPD) concerning the best solutions found for each instance, computed according to Equation (7-1). $TWCT^{Sol}$ denotes the total weighted completion for a given solution of a specific instance, and $TWCT^{Best}$ designates the total weighted completion time regarding the best solution found for the same instance in a given experiment.

$$RPD = \frac{TWCT^{Sol} - TWCT^{Best}}{TWCT^{Best}} \times 100. \tag{7-1}$$

7.2.1 Results and Calibration

Before presenting the results, we set the input parameters of the `IG-RG`, following a two-phase tuning strategy introduced by Ropke and Pisinger (2006). First, parameters are defined in a trial-and-error phase during the algorithm development. Then, a fine-tuning is executed in each parameter individually within the improvement phase, considering pre-defined possible values. We executed the fine-tuning in the following order: (1) Initial temperature parameter for the simulated annealing (δ_1), with values ranging from 0.3 to 0.7 and a step size of 0.1; (2) Final temperature parameter for the simulated annealing (δ_2) with the possible values of 10^{-3}, 10^{-4}, 10^{-5}, and 10^{-6}. (3) Perturbation parameter (ε) with values ranging from 0.05 to 0.20 and a step size of 0.05; (4) Solution restore parameter (λ), within the same range of values tested for ε. (5) Penalty update factor when infeasible solutions are reached (ρ^+), defined within the range 0.10 to 0.25, with a step size of 0.05; (6) Penalty update factor when feasible solutions are reached (ρ^-) defined within the range of 0.05 to 0.20, with a step size of 0.05. We performed ten independent runs on each instance with $\eta = 2500$ iterations. The number of iterations was defined during the trial-and-error phase. Table 7.1 presents the final parameter values. These values were set according to the average relative percentage deviation ($\overline{RPD}$) concerning the best solutions achieved during the parameterization.

Table 7.1: Final parameter values for the proposed algorithm.

Parameter	Description	Domain	Value
δ_1	Initial temperature definition parameter for the simulated annealing criterion	[0, 1]	0.6
δ_2	Final temperature definition parameter for the simulated annealing criterion	[0, 1]	10^{-5}
ε	Proportion of the total number of operations to destroy at each iteration	[0, 1]	0.15
λ	Restore solution parameter	[0, 1]	0.1
ρ^+	Parameter to update the penalty factor when infeasible solutions are accepted	[0, 1)	0.20
ρ^-	Parameter to update the penalty factor when feasible solutions are accepted	[0, 1)	0.05

In the following, we evaluate the impact on the solution quality by removing each feature of the `IG-RG` algorithm. Table 7.2 shows the average relative percentage deviation ($\overline{RPD}$), standard deviation of the RPDs (SD), and the average computational time ($\overline{Time}$), in seconds, for each configuration. Each configuration corresponds to the disabling of one of the following features: (LS) RVND local search; (SA) Simulated annealing acceptance criterion;

(DR) Destroy and Repair steps; (Inf.) Infeasibility strategy; (Rest.) Solution restore strategy. It is worth mentioning that the configuration without the destroy and repair steps, we replace these steps by a regular perturbation step, following an Iterated Local Search (ILS) metaheuristic structure, in which we randomly select one neighborhood, and performs $d = \lceil \varepsilon o \rceil$ random moves. One can note that the local search is the most relevant and time-consuming feature. The average computational time reduces when the algorithm disregards the diversification strategies (infeasibility strategy and simulated annealing). However, the deviation reduces when we consider these strategies with a small addition of time. One can note that the configuration with all features combined generates the smallest $\overline{RPD}$ and standard deviation (SD), indicating that they all contribute to the algorithm's performance.

Table 7.2: Average RPD and computational time with the removal of each algorithm's feature.

Config.	LS	SA	DR	Inf.	Rest.	$\overline{RPD}$	SD	$\overline{Time}$
No LS		•	•	•	•	2.68	2.38	0.15
No SA	•		•	•	•	0.22	0.42	7.70
No DR	•	•		•	•	0.37	0.90	12.86
No Inf.	•	•	•		•	0.19	0.31	7.69
No Rest.	•	•	•	•		0.18	0.32	8.20
Complete	•	•	•	•	•	0.15	0.26	8.18

In the next analysis, shown in Figure 7.6, we evaluate the trade-off between the solution quality in terms of the $\overline{RPD}$ and the average computational time, regarding the number of iterations considered. We ran the experiments with the number of iterations (η) varying from 500 to 10000 with a step size of 500. One can note that the average time grows linearly as the number of iterations increases. We present the $\overline{RPD}$ with a 95% confidence interval. Note that the $\overline{RPD}$ decreases as the number of iterations grows, reducing faster up to 4000 iterations. From 4000 to 4500, the method stabilizes. It returns to a continuous reduction of the $\overline{RPD}$, but slower, from 4500 to 7000 iterations. Then, from 7000 iterations, the method is practically stable until reaching the maximum number of iterations tested (10000), indicating that it converges to a $\overline{RPD}$ around 0.08%.

In the last analysis, we compare the `IG-RG` running for 2500, 4500, and 7000 iterations, following the previous analysis, with the matheuristics presented in Chapter 6 in terms of the RPD, computational time, and capacity of reaching the best-found solution.Table 7.3 presents the results in terms of the average (Avg), maximum (Max) and standard deviation (SD) for the RPDs and

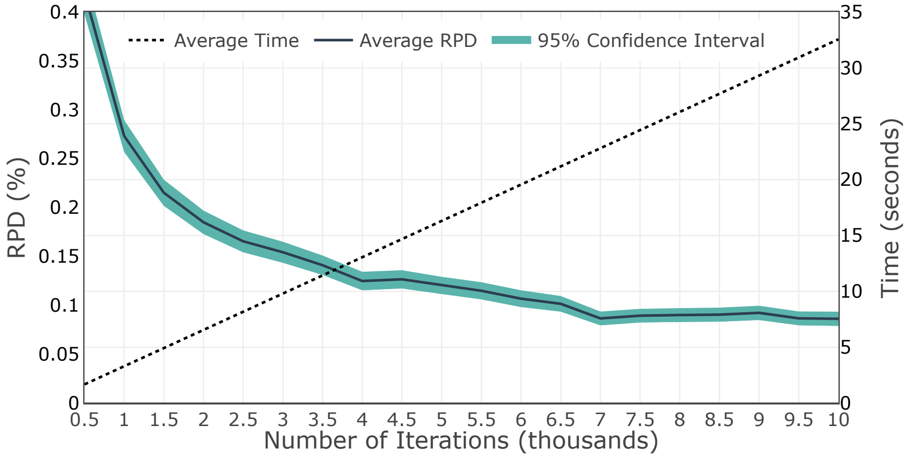


Figure 7.6: Average RPD and average computational time with different number of iterations (Mecler et al. 2021).

computational times, the percentage of instances (%Inst) and runs (%Runs) in which each algorithm achieved the best solution, and the percentage of instances in which the best solution is uniquely found by a given approach (%Unique). We included the number of iterations in the algorithm's name.

Table 7.3: Comparison between the iterated greedy algorithm and the best methods in the literature for this set of instances.

Algorithm	RPD (%)			Time (seconds)			Best Solution Achieved		
	Avg	Max	SD	Avg	Max	SD	%Inst.	%Run	%Unique
`GRASPMath`$_3$	0.79	20.57	1.28	464.7	2677.4	529.8	62.50	40.56	0.00
`ILS-Math`$_3$	0.74	6.62	0.96	411.7	2735.2	503.5	58.33	37.64	1.39
`IG-RG-2500`	0.15	2.03	0.28	8.2	26.8	9.7	73.61	61.25	0.00
`IG-RG-4500`	0.11	1.47	0.23	14.7	47.3	17.5	83.33	64.86	4.17
`IG-RG-7000`	0.07	1.23	0.16	22.8	75.3	27.1	94.44	70.00	19.44

One can note that the `IG-RG` is superior in all criteria independently of the number of iterations. Even when running for 2500 iterations (`IG-RG-2500`), its worst-case RPD (Max) remains around 2%, with a low standard deviation (0.28), showing its consistency within the different runs. Moreover, even when the number of iterations grows, as in the case of the `IG-RG-7000`, the average computational time is at least 94% lower than the matheuristic ones. The algorithm reached the best solution in 94.44% of the instances and 70% of the runs when running for 7000 iterations, providing new best solutions (upper bound) for the benchmark set of instances. Complete results are shown in Appendix E.1.

7.3 Discussion

This chapter introduced an Iterated Greedy algorithm, using a random destroy operator and a greedy repair operator to solve the PLSVSP. The results show that the IG algorithm outperforms the matheuristics in computational time and solution quality, providing new upper bounds for some instances. The study reinforces the applicability of iterated greedy algorithms in solving combinatorial optimization problems even when a complicated problem inspired by a real context is considered. The idea of destroying and repairing the solution is a simple yet powerful and efficient technique, as we confirmed in our experiments.

8
Iterated Greedy Simheuristic with Embedded Monte Carlo Simulation for the stochastic PLSVSP

In this chapter, we present a simheuristic to solve the PLSVSP with stochastic processing times and release dates for the operations. The method embeds a Monte Carlo Simulation into an IG approach to identify good stochastic solutions. With a small number of replications, short simulations are performed during the algorithm to identify and store the most promising solutions, according to a chosen statistic, in a pool with the best stochastic solutions. In the end, a long simulation, with a higher number of replications, is performed on all solutions in the pool to generate random sample observations and provide more statistical information about the solutions.

The general concept of the simheuristic is based on Grasas et al. (2016). However, instead of using an Iterated Local Search (ILS) metaheuristic, we use an Iterated Greedy (IG) approach due to the successful application of this algorithm for solving the deterministic PLSVSP (Chapter 7) and other machine scheduling problems Ruiz and Stützle (2007), Fanjul-Peyro and Ruiz (2010), Lee (2017), Ruiz et al. (2019). The method, named `SimIG`, combines the simulation step with the IG approach described in Chapter 7. Furthermore, unlike regular simheuristics in the literature, in our approach, we do not define the number of replications to be performed during the simulation steps. The use of a fixed number of replications during the simulation can lead to two issues. On the one hand, a low number of replications does not guarantee that the sample's expected objective value represents a given solution properly. On the other hand, a high number of replications might result in computational overhead for the method. To overcome these issues, we incorporate a strategy proposed by González-Neira et al. (2019) of using the confidence interval's error around the mean, from the available observations, to control the number of simulation replications. Thus, after each replication, confidence intervals at a given confidence level are calculated, and the algorithm computes the interval error. The procedure stops when the calculated error falls below the desired pre-established limit.

The general flowchart of the IG simheuristic is shown in Figure 8.1. The method is divided into three parts. The initialization, in which the algorithm constructs a solution, executes the destroy and repair steps and the local search, runs a short simulation, and initializes the stochastic pool. The main loop, in which the method runs the destroy and repair phases iteratively and checks whether the solution should be accepted, simulated, and included in the stochastic pool. And the termination, in which a long simulation is performed on each solution within the stochastic pool.

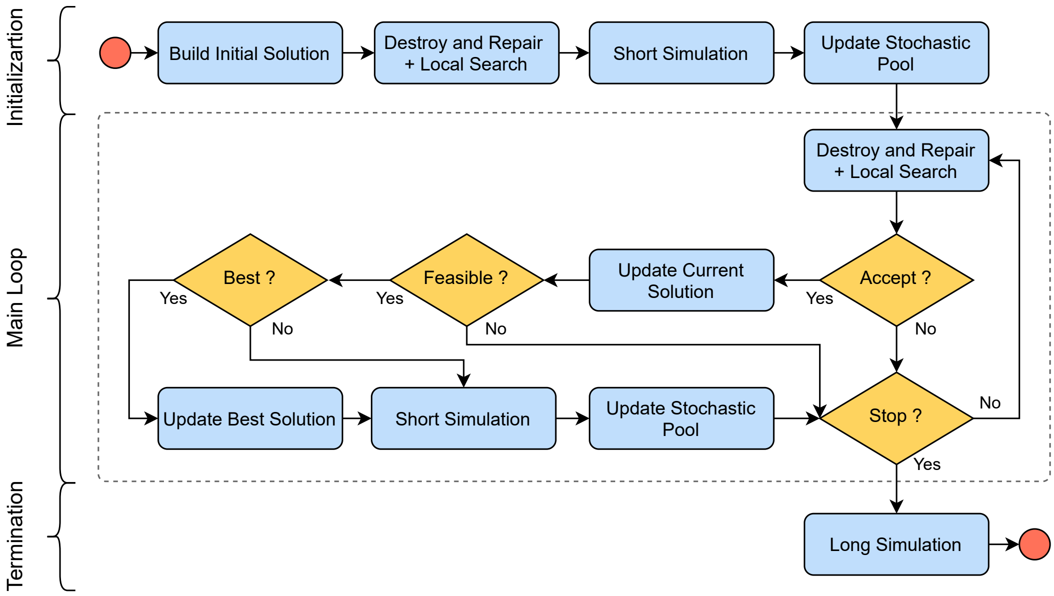


Figure 8.1: General simheuristic flowchart.

8.1 Simheuristic pseudo-code and description

The `SimIG` pseudocode is shown in Algorithm 11. Eight parameters must be set to run the algorithm: the number of iterations (η), the size of the stochastic pool ($size$), the confidence levels for the short and long simulations (ϱ_s and ϱ_ℓ), the error limit for the short and long simulations ($\tilde{\varepsilon}_s$ and $\tilde{\varepsilon}_\ell$), and the variance levels for the stochastic parameters (δ_p and δ_r). During its execution, the algorithm uses $\mathtt{s}$ to represent the current solution, $\mathtt{s}'$ for candidate solutions, and $\mathtt{s}^*$ to indicate the best-found deterministic solution. Moreover, $f(.)$ is a function that returns the deterministic total weighted completion time of a given solution. And, the set $\mathcal{P}$ represents the pool of the best stochastic solutions. In the first part of the algorithm (Lines 1–4), an initial solution is created and improved by a local search phase. This solution is then simulated, and the stochastic pool $\mathcal{P}$ is initialized containing only it. The method's main loop (Lines 5–20) is repeated for η iterations and consists of executing a destroy and repair phase followed by a local search procedure and an acceptance evaluation. Solutions are restored to the best-found after a pre-defined number of consecutive iterations without improvement (Line 19). One of the main differences from the IG algorithm (Chapter 7) is to consider a pool of stochastic solutions, using a short simulation step to identify and store feasible promising solutions (Line 12). The stochastic pool is updated by including these solutions or not (Line 13). After the main loop, a long simulation is performed on the solutions of the stochastic pool (Lines 21–23) to get more statistical information. The local search, destroy and repair phases, infeasibility strategy, and restore strategy are the ones discussed in Chapter 7. The `Simulation` and `PoolUpdate` procedures are detailed in the next sections.

Algorithm 11: `SimIG`

Input : Number of iterations (η); Size of the stochastic pool ($size$); Confidence level for the short simulation (ϱ_s); Error limit for the short simulation ($\tilde{\varepsilon}_s$); Confidence level for the long simulation (ϱ_ℓ); Error limit for the long simulation ($\tilde{\varepsilon}_\ell$); variance level for the processing times (δ_p) and release dates (δ_r); minimum number of replications within the simulation steps (ϕ).

Output: The pool $\mathcal{P}$ with the best stochastic solutions.

1 $\texttt{s} \leftarrow$ `Constructive()`;
2 $\texttt{s}^* \leftarrow \texttt{s} \leftarrow$ `LocalSearch(s)`;
3 `Simulation`$(\varrho_s, \tilde{\varepsilon}_s, \delta_p, \delta_r, \phi, \texttt{s})$;
4 $\mathcal{P} \leftarrow \{\texttt{s}\}$;
5 **for** η *iterations* **do**
6     $\texttt{s}' \leftarrow$ `Repair(Destroy(s))`;
7     $\texttt{s}' \leftarrow$ `LocalSearch`$(\texttt{s}')$;
8     $is\_feasible \leftarrow$ `Feasible`$(\texttt{s}')$;
9   **if** `Accept`$(\texttt{s}', \texttt{s})$ **then**
10         $\texttt{s} \leftarrow \texttt{s}'$;
11   **if** $is\_feasible$ **then**
12   `Simulation`$(\varrho_s, \tilde{\varepsilon}_s, \delta_p, \delta_r, \phi, \texttt{s})$;
13             $\mathcal{P} \leftarrow$ `PoolUpdate`$(\mathcal{P}, size, \texttt{s})$;
14   **if** $f(\texttt{s}) < f(\texttt{s}^*)$ **then**
15                 $\texttt{s}^* \leftarrow \texttt{s}$;
16   **end**
17   **end**
18   **end**
19     $\texttt{s} \leftarrow$ `Restore`$(\texttt{s}^*, \texttt{s})$;
20 **end**
21 **for** $\texttt{s} \in \mathcal{P}$ **do**
22   `Simulation`$(\varrho_\ell, \tilde{\varepsilon}_\ell, \delta_p, \delta_r, \phi, \texttt{s})$;
23 **end**
24 **return** $\mathcal{P}$;

8.1.1 Simulation

The `Simulation` procedure receives a solution `s` to assess its quality in the stochastic environment, generating a sample set $\mathcal{R}_\texttt{s}$ of several simulation replications values. The sample provides more statistical information about solution $\mathtt{s}$. As mentioned before, the total number of replications is not known *a priori* but depends on an error limit defined for the confidence interval around the mean at a given confidence level. However, we define a minimum number of replications (ϕ) to ensure a minimum sample size to compute the statistics.

The pseudocode of the `Simulation` procedure is shown in Algorithm 12. First, the set $\mathcal{R}_{\mathtt{s}}$ of total weighted completion time observations is initialized as empty and the sample-set error as infinity (Lines 1–2). Then, in each replication, the `MonteCarlo` procedure generates random values for the stochastic parameters (Lines 6–7), generating new processing times p'_i and release dates r'_i for each operation $O_i \in \mathcal{O}$, following the defined probability distributions and variance levels (δ_p and δ_r). According to the defined schedule, the `Evaluate` procedure calculates the new completion times for operations and jobs and the total weighted completion time (v) of the current replication (Line 8). Batches are anticipated whenever possible and delayed if necessary during the evaluation. The set $\mathcal{R}_{\mathtt{s}}$ is updated by including the new observation value v, and the mean and standard deviation of the sample are computed by the `Statistics` procedure (Lines 9–10). Then, the `ConfIntervalWidth` procedure builds a confidence interval for the mean, considering the total number of replications and the desired confidence level ϱ, returning the interval *width* (Line 11). The simulation process stops when the error ε (computed in Line 12) is less than or equal to a pre-defined desired error limit $\tilde{\varepsilon}$. Moreover, a minimum number of replications (ϕ) is also required to stop the procedure. More statistical information can be obtained with the sample of stochastic total weighted completion times such as the sample standard deviation, the value at risk (VaR), the expected shortfall (a.k.a. conditional value at risk – CVaR), and others. The expected shortfall (CVaR_α) is a risk metric that measures the average of the worse values of a given distribution beyond a defined quantile α, known as the value at risk (VaR_α). The CVaR has been used in many financial and engineering risk management works (Street 2010).

To help the readers clarify how the schedule is affected by the uncertainties, we depict in Figure 8.2 an example of one simulation replication with stochastic values for the processing times (p_i) and release dates (r_i) for the operations, considering four operations scheduled on a single machine. The original schedule refers to the one of machine M_2 in the example shown in Figure 3.5. The original and simulated values for the stochastic parameters are shown in Table 8.1. Operations O_9 and O_8 compose the first batch, while the second and third batches are composed by operations O_{11}, and O_5, respectively. Note that the sequence of operations and the batch compositions are the same in the original and simulated schedules. In the original sched-

Algorithm 12: `Simulation`

Input : Confidence level (ϱ); Error limit ($\tilde{\varepsilon}$); variance levels for the processing times (δ_p) and release dates (δ_r); minimum number of replications (ϕ); A solution $\mathtt{s}$.

$\mathcal{R}_{\mathtt{s}} \leftarrow \emptyset$;
$\varepsilon \leftarrow \infty$;
$replications \leftarrow 0$;
while $\varepsilon > \tilde{\varepsilon}$ **or** $replications \leq \phi$ **do**
  $replications \leftarrow replications + 1$;
  $p'_i \leftarrow$ `MonteCarlo(`P_i, δ_p`)`, $\forall O_i \in \mathcal{O}$;
  $r'_i \leftarrow$ `MonteCarlo(`R_i, δ_r`)`, $\forall O_i \in \mathcal{O}$;
  $v \leftarrow$ `Evaluate(`$\mathtt{s}, p', r'$`)`;
  $\mathcal{R}_{\mathtt{s}} \leftarrow \mathcal{R}_{\mathtt{s}} \cup \{v\}$;
  $(\mu, \sigma) \leftarrow$ `Statistics(`$\mathcal{R}_{\mathtt{s}}$`)`;
  $width \leftarrow$ `ConfIntervalWidth(`$\mu, \sigma, \varrho, replications$`)`;
  $\varepsilon \leftarrow width/\mu$;
end

ule, the first batch starts in $t = 9$ due to the release date of operation O_9 ($r_9 = 9$). The replication defines the release dates for operations O_8 and O_9 as zero ($r'_8 = r'_9 = 0$), allowing to anticipate the batch starting time to $t = 0$. One can note that the second batch was also anticipated, changing its starting time from $t = 43$ to $t = 38$. However, idleness is generated due to the new release date of operation O_{11} ($r'_{11} = 38$). The new processing time of operation O_{11} ($p'_{11} = 10$) forces a delay on starting the third batch, changing its starting time from $t = 52$ to $t = 53$. The completion times for operations are shown in both schedules, pointing that operations O_9 and O_8 were anticipated while operations O_{11} and O_5 were delayed.

Table 8.1: Original and simulated values for the processing times and release dates used in the simulation example, considering one replication.

Parameters		Operation (O_i)			
		O_5	O_8	O_9	O_{11}
Original	p_i	4	8	17	4
	r_i	18	5	9	3
Simulated	p'_i	7	7	15	10
	r'_i	20	0	0	38

8.1.2 Updating the Stochastic Pool

The `PoolUpdate` procedure tests whether the pool $\mathcal{P}$ of best stochastic solutions should include a candidate solution $\mathtt{s}$ or not, considering the pool size ($size$) and the value v of the candidate solution. Any statistics on the

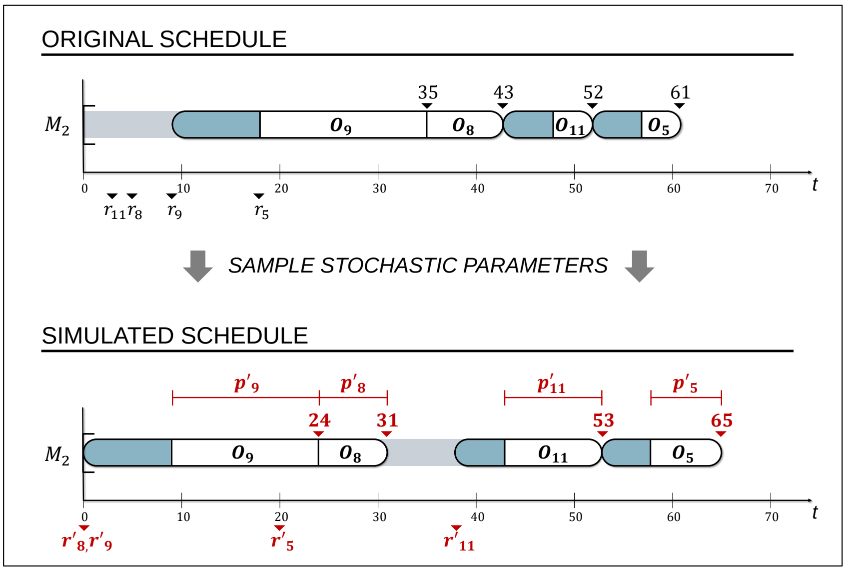


Figure 8.2: Simulation example with one replication for a single ship schedule.

distribution of stochastic total weighted completion times obtained after the simulation step can be used to evaluate the solutions' values, including the mean, standard deviation, VaR, CVaR, and others.

The pseudocode of the `PoolUpdate` procedure is shown in Algorithm 13. First, the algorithm checks whether the candidate solution $\mathtt{s}$ is already in the pool $\mathcal{P}$ (Line 1). If not, the algorithm retrieves the value of the worst solution in the pool (v') and saves the respective solution $\mathtt{s}'$ (Lines 2–6). Note that the worst solution and its value are only identified when the pool is full. Then, the algorithm saves the value v of the candidate solution $\mathtt{s}$ (Line 7). If the value v of the candidate solution $\mathtt{s}$ is better than v' (Line 8), $\mathtt{s}$ is included in the pool (Line 9). Finally, the algorithm checks if the maximum number of solutions in the pool has been surpassed, removing solution $\mathtt{s}'$, if true (Lines 10–12).

8.2 Computational Experiments

In this section, we present and discuss the experiments conducted within the PLSVSP benchmark set of 72 instances to evaluate the performance of the `SimIG` algorithm. We coded the simheuristic using C++ language and performed the experiments on an Intel i7-9700 CPU 3.0GHz machine with 16GB of RAM and running Linux. We ran the simheuristic ten times in each experiment using one single thread with different seeds. The simheuristic uses a confidence level of 95% during the short simulations with an error limit of 1%. These values for the long simulation are 99% and 0.1%, respectively. We set the minimum number of replications as 30. Complete results are available

Algorithm 13: `PoolUpdate`

Input : Pool of stochastic solutions ($\mathcal{P}$); Maximum number of solutions in the pool ($size$); A candidate solution ($\mathtt{s}$).
Output: Updated stochastic pool.

if $\mathtt{s} \notin \mathcal{P}$ **then**
  $\mathtt{s}' \leftarrow \emptyset$;
  $v' \leftarrow \infty$;
  if $|\mathcal{P}| = size$ **then**
    $(\mathtt{s}', v') \leftarrow$ the worst solution in $\mathcal{P}$ and its value;
  end
  $v \leftarrow$ value of solution $\mathtt{s}$;
  if $v < v'$ **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathtt{s}\}$;
    if $|\mathcal{P}| > size$ **then**
      $\mathcal{P} \leftarrow \mathcal{P} \setminus \{\mathtt{s}'\}$;
    end
  end
end
return $\mathcal{P}$;

in Appendix F.1. We use the Boost C++ Libraries[1] to compute the statistics and build the confidence intervals on the mean. We use the random library from the C++ standard library for the Monte Carlo Simulation to produce the random values for the stochastic parameters according to the Log-Normal distribution.

As mentioned before, the simheuristic can use any statistics over the simulated sample of the objective values during its execution to decide which solutions to include in the pool of the best stochastic solutions. In our experiments, we consider two `SimIG` variants. The `SimIG-Exp`, which uses the expected objective value as the selection criterion, and the `SimIG-CVaR`, using the $\text{CVaR}_{95\%}$ as the selection criterion. We also consider the regular Iterated Greedy algorithm, named `IG`, to compare with the `SimIG` algorithms. The `IG` is the `SimIG` without the short simulation steps, returning the best-found deterministic solution. However, the long simulation step is also executed in the `IG` to evaluate the algorithm's solutions submitted to the stochastic environment.

In our problem, each solution $\mathtt{s}$ represents a schedule for the machines. We focus our analyzes on three metrics given by each solution: (1) the deterministic total weighted completion time $f(\mathtt{s})$ of a solution $\mathtt{s}$; (2) the expected total weighted completion time $E[\mathtt{s}]$ of a solution $\mathtt{s}$ after the long simulation step; (3) the expected shortfall at 95% $CVaR(\mathtt{s})$ regarding the sample of stochastic total weighted completion times obtained after the long

[1]http://www.boost.org, last accessed 2021-01-29.

simulation of a solutions $\mathtt{s}$.

All solution evaluations are done regarding the Relative Percentage Deviation (RPD) to the best deterministic solutions, to allow comparing the results from different instances. Thus, we consider three RPDs in our analyses, the $D\text{-}RPD(\mathtt{s})$ (Equation 8-1), the $E\text{-}RPD(\mathtt{s})$ (Equation 8-2), and the $C\text{-}RPD(\mathtt{s})$ (Equation 8-3). The first one computes the RPD from the deterministic objective value $f(\mathtt{s})$ of a solution $\mathtt{s}$ to the deterministic value $f(best)$ of the best solution ($best$), while the other two do the same but using the expected objective value $E[\mathtt{s}]$ and expected shortfall at 95% $CVaR(\mathtt{s})$ of solution $\mathtt{s}$. It is worth mentioning that the best deterministic solutions ($best$) are the ones found by the `IG` algorithm.

$$D\text{-}RPD(\mathtt{s}) = \frac{f(\mathtt{s}) - f(best)}{f(best)} \times 100, \tag{8-1}$$

$$E\text{-}RPD(\mathtt{s}) = \frac{E[\mathtt{s}] - f(best)}{f(best)} \times 100, \tag{8-2}$$

$$C\text{-}RPD(\mathtt{s}) = \frac{CVaR(\mathtt{s}) - f(best)}{f(best)} \times 100. \tag{8-3}$$

8.2.1 Experiments with three variance levels for the stochastic parameters

In the first experiment, we compare the performance of `IG` and `SimIG-Exp` for three variance levels (low, medium, and high variance) of the stochastic parameters. For the low variance scenario, we define $\delta_p = \delta_r = 0.5$, while for the medium and high variance levels these values are 2.0 and 5.0, respectively. These are the same values used in Juan et al. (2014) to generate stochastic processing times for a flow shop scheduling problem. This analysis only considers the best solutions found by each algorithm. Thus, the best solution of the `SimIG-Exp` algorithm is the one with the best expected total weighted completion time among 100 generated solutions (ten independent runs and ten solutions in each run's pool). The best solution for the `IG` is the one with the best deterministic total weighted completion time among the ten independent runs.

Table 8.2 summarises the results for each variance level (low, medium, and high) grouped by the number of operations to schedule (o) and the number of machines available (m). Average values for the $E\text{-}RPD$ ($\overline{E\text{-}RPD}$) and computational times ($\overline{Time}$), in seconds, are shown for each algorithm. The expected objective values are obtained after running the long simulation on the best solution found by the `IG` and the best stochastic solution found by the `SimIG-Exp`. We also included the average values for the $D\text{-}RPD$ ($\overline{D\text{-}RPD}$) for the `SimIG-Exp` algorithm. The table suppresses this indicator for the `IG` algorithm since its solutions are the references for computing the $D\text{-}RPD$ values (i.e., $D\text{-}RPD = 0.0$ for all instances). The table also shows the percentage difference (Gap) between the expected RPDs ($E\text{-}RPD$) of the algorithms. Finally, we included the average number of short simulations executed ($\overline{SSim}$) and the average number of replications for the short and long simulation steps ($\overline{SRep}$ and $\overline{LRep}$) from the `SimIG-Exp` algorithm.

Table 8.2: Results summary for each algorithm grouped by the number of operations and machines considering three variance levels.

Variance Level	o-m	`IG`		`SimIG-Exp`						
		$\overline{E\text{-}RPD}$	$\overline{Time}$	$\overline{D\text{-}RPD}$	$\overline{E\text{-}RPD}$	$\overline{Time}$	Gap	$\overline{SSim}$	$\overline{SRep}$	$\overline{LRep}$
Low	15-4	2.88	0.63	0.17	2.62	1.51	-9.27	1681	56	9710
	15-8	3.28	0.64	0.09	2.61	2.00	-20.44	1608	79	14738
	25-4	3.20	2.10	0.13	2.69	2.82	-15.82	1570	38	5967
	25-8	4.45	2.50	0.41	3.85	3.33	-13.62	1378	41	7008
	50-4	3.20	22.16	0.19	2.94	23.09	-8.19	1197	31	2717
	50-8	4.35	21.76	0.41	3.88	22.88	-10.77	1109	31	3382
Medium	15-4	8.26	0.80	1.23	7.35	5.06	-11.01	1681	243	42225
	15-8	9.15	0.87	0.65	7.81	6.63	-14.68	1608	319	58574
	25-4	8.80	2.24	0.55	7.77	5.72	-11.70	1570	152	26536
	25-8	11.74	2.68	1.01	10.15	6.60	-13.55	1378	178	31463
	50-4	8.32	22.20	0.60	7.67	25.05	-7.86	1197	67	12100
	50-8	11.41	21.84	0.73	10.24	25.24	-10.22	1109	86	15627
High	15-4	15.78	1.21	1.96	13.87	13.21	-12.09	1681	684	115940
	15-8	17.31	1.40	1.28	15.36	17.03	-11.27	1608	860	155802
	25-4	16.74	2.56	1.00	15.03	12.48	-10.21	1570	429	74607
	25-8	21.73	3.10	1.56	19.14	14.80	-11.90	1378	522	90923
	50-4	15.49	22.52	0.98	14.28	29.67	-7.84	1197	188	33411
	50-8	21.38	22.18	1.22	19.50	31.89	-8.78	1109	257	45843

Positive values of $\overline{D\text{-}RPD}$ indicate worse deterministic total weighted completion times for solutions generated by the `SimIG-Exp`. However, the values of $\overline{E\text{-}RPD}$ are lower for `SimIG-Exp` compared to `IG`, indicating `SimIG-Exp` superiority when its solutions are submitted to the stochastic environment. Note that the difference between the $\overline{E\text{-}RPD}$ values can be noted by the Gap column, which indicates a reduction of at least 3.39% and up to 16.52% for the `SimIG-Exp` algorithm. One can notice that the differences grow as the variance level increases. Besides, the values of $\overline{D\text{-}RPD}$ are also higher as the variance level increases, highlighting the advantage of simheuristics in these cases, given that the solution (i.e., the schedule defined for the machines) generated by the `SimIG-Exp` diverges more from the solution built by the `IG` algorithm. Also, by analyzing the average computational times, it is noticed that the overhead caused by the simulation in `SimIG-Exp` is low, which makes the simheuristic algorithms competitive against regular metaheuristics in generating solutions for stochastic problems. The average computational times of the `IG` are independent of the variance level since the method does not have a short simulation step.

Regarding the simulation step, one can note that both the number of simulation calls ($\overline{SSim}$) and the number of replications ($\overline{SRep}$) performed in the short simulation decrease as the instance size grows. This behavior occurs because an extreme value in one stochastic parameter impacts proportionally more the objective function value when a smaller number of operations are scheduled. That is, if an operation has its completion time increased due to uncertainty, it would be one operation among 15 affecting the schedule for smaller instances and one between 50 for larger instances. Besides, the lower objective values of solutions with fewer operations result in a higher proportional increment when the solutions are affected by the stochastic environment. As expected, it is evident that the number of replications increases when the variance level grows for both short and long simulation steps ($\overline{SRep}$ and $\overline{LRep}$). This analysis reinforces the advantage of using the iterative mechanism of building confidence intervals during the simulation to define the number of replications since this number is instance and variance level dependent.

Figure 8.3 depicts the $E\text{-}RPD$ distributions for the best solutions found by the `IG` and `SimIG-Exp` algorithms within the three variance levels considering the complete set of instances. The zero-line corresponds to the best deterministic solution. As mentioned before, this value is the reference used to assess the impact of uncertainties when the solutions are submitted to the stochastic environment. Not surprisingly, the consideration of uncertainties disrupts the schedule, affecting the expected values of the solutions. Three main aspects can be noted from the $E\text{-}RPD$ distributions: (1) the higher the uncertainty, the larger will be the deviations; (2) the expected values are bounded by the best deterministic total weighted completion time (zero line), meaning that even when the disruptions are low, the expected values for the objective function are significantly impacted; (3) with increasing the uncertainty, the differences between the distributions for the best solutions found by the algorithms are accentuated.

To validate the previous analysis, we performed a statistical evaluation of

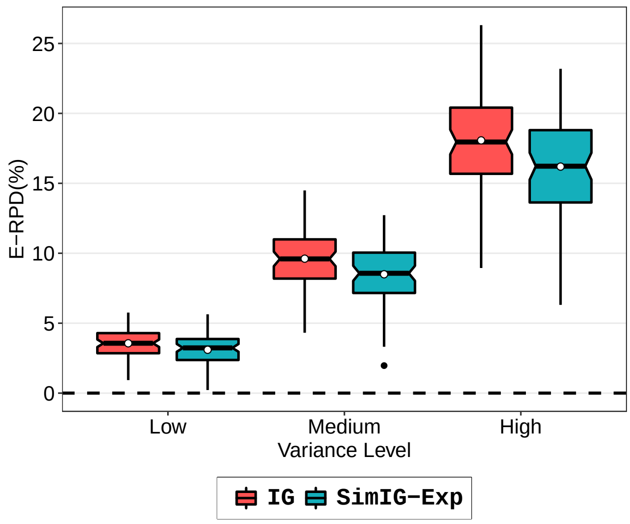


Figure 8.3: Boxplot of the $E\text{-}RPD$ distributions for each algorithm within three variance levels.

the $E\text{-}RPD$ distributions of each algorithm's best solutions for each variance level. Before running a statistical test, we confirmed that both samples follow normal distributions according to the Shapiro-Wilk test. Then, we ran the Analysis of Variance (ANOVA) with the Tukey HSD (honestly significant difference), at a confidence level of 0.05, to compare the algorithms' $E\text{-}RPD$ distributions. The tests indicated a statistically significant difference between the algorithms in all scenarios. The p-values are below 0.01 in all cases. The statistical evaluation corroborates the discussion about the difference between the $E\text{-}RPD$ distributions and points out a significant advantage for simheuristics even with low variance level scenarios.

8.2.2
Sensitivity analysis regarding the variance levels

This section evaluates the sensitivity of the stochastic parameters' variance levels, only considering the `SimIG-Exp` algorithm. First, we evaluate the impact of each stochastic parameter individually. Thus, three scenarios are considered: (1) stochastic release dates within three variance levels (In this scenario, $\delta_p = 0$ in all cases); (2) stochastic processing times within three levels of variance (In this scenario, $\delta_r = 0$ in all cases); (3) stochastic release dates and processing times within three variance levels (In this scenario $\delta_p = \delta_r$ in all cases). One can note that Scenario 3 corresponds to the $E\text{-}RPD$ distributions for the `SimIG-Exp` algorithm depicted in Figure 8.3. Figure 8.4 shows the $E\text{-}RPD$ distributions for the `SimIG-Exp` for the three proposed scenarios. Again, the zero-line corresponds to the best deterministic solution.

Based on the boxplots, one can note a higher disturbance on the expected

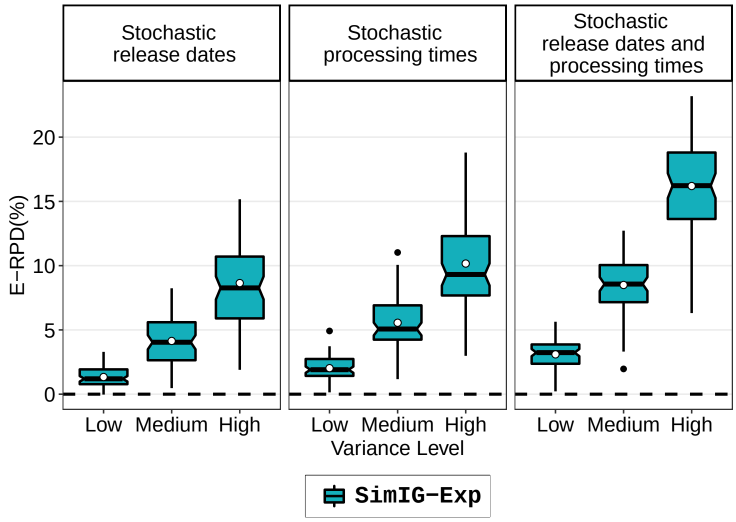


Figure 8.4: Boxplot of the *E-RPD* distributions for the `SimIG-Exp` algorithm isolating the impact of the stochastic parameters.

objective function values with stochastic processing times than stochastic release dates. This behavior can be explained by the lower values for the release dates, thus having a higher probability of impacting the first scheduled batches. Despite the discrepancy between the distributions when we isolate each stochastic parameter's impact, it is evident that the combination of uncertainty in both parameters causes a more significant disruption in the schedules, making the problem more challenging to be addressed.

In the next experiment, we evaluate the *E-RPD* distributions of the best solutions generated by the `SimIG-Exp` algorithm for all combinations between the variance levels for both stochastic parameters. To indicate the specified variance level of each parameter, we introduce the following labels: (N) no variance; (L) low variance; (M) medium variance; and (H) high variance. For instance, the label MH means that the scenario considers medium variance for the stochastic processing times and high variance for the stochastic release dates, that is, $\delta_p = 2.0$ and $\delta_r = 5.0$. Figure 8.5 shows the *E-RPD* distributions, with the horizontal axis sorted according to the distributions' mean values, for a better visualization.

Note that configuration MM is centered in the graph, dividing the scenarios between those with lower impact and higher impact on the solutions' expected objective values. All combinations with at least one high variance level are located on the right side of the graph (scenarios with higher average values for the *E-RPD*). The increase in the mean of the distributions is evident when both parameters assume the same variance level. Additionally, an almost linear increase is noticed until the graph reaches the ML configuration, then the increase is steeper between the ML and the MM configurations. The same

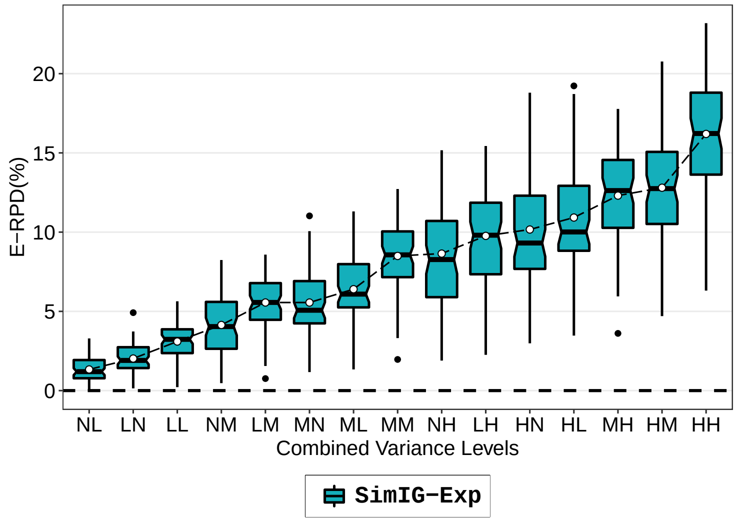


Figure 8.5: Boxplot of the *E-RPD* distributions for the `SimIG-Exp` algorithm with mixed variance levels among the stochastic parameters.

behavior occurs from the MM configuration on, it is almost linear up to the HM configuration, and then there is a higher increase to the HH configuration. This progression highlights the impact of considering stochastic processing times and release dates in the problem.

8.2.3 Risk Analysis

In the next experiment, we evaluate the performance of the `SimIG-CVaR` against the `IG` and `SimIG-Exp` algorithms, including the risk perspective to the solutions' analysis. We also consider the best solutions regarding the defined selection criterion for the `SimIG` variants in this analysis. In addition to the expected values, we are interested in analyzing the $\text{CVaR}_{95\%}$ of the total weighted completion times observations of the defined solutions. Figure 8.6 depicts the *E-RPD* and *C-RPD* distributions for the best solutions found by the `IG`, `SimIG-Exp`, and `SimIG-CVaR` algorithms within the three variance levels, considering the complete set of instances. The zero-line corresponds to the best deterministic solution.

As noted in the first analysis, the superiority of the `SimIG-Exp` algorithm upon the `IG` algorithm is evident. In this analysis it is noticed that the `SimIG-CVaR` variant is also better than the `IG` algorithm in both criteria analyzed (*E-RPD* and *C-RPD*). Moreover, although worse in the analysis of the *E-RPD*, the advantage of `SimIG-CVaR` over `SimIG-Exp` is evident when the criterion considered is the $\text{CVaR}_{95\%}$. This behavior was expected, given that this criterion was used during optimization. However, an interesting result is to realize that the reduction in the $\text{CVaR}_{95\%}$ values when optimized by the

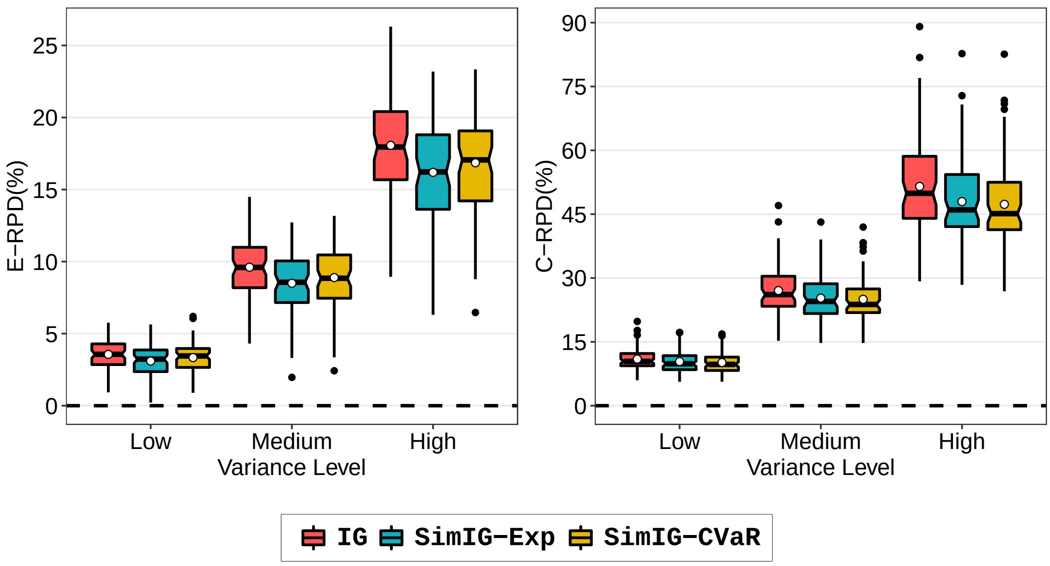


Figure 8.6: Boxplot of the *E-RPD* and *C-RPD* distributions for each algorithm within three variance levels.

`SimIG-CVaR` algorithm does not cause major impacts on the expected values.

The previous analysis highlights the challenge faced by planners when choosing a schedule to be followed. We picked a large instance (50 operations), within a high variance scenario ($\delta_p = \delta_r = 5.0$), to analyze the trade-off between the *E-RPD* and the *C-RPD* indicators, to reinforce the risk analysis. Figure 8.7 depicts the described trade-off for all solutions generated by each one of the algorithms (`IG`, `SimIG-Exp`, and `SimIG-CVaR`) regarding the ten independent runs. Thus, 100 solutions are considered for the `SimIG` variants and 10 solutions for the `IG` algorithm. In this analysis, we highlight the set of dominant solutions (Pareto frontier). Among them, three were generated by the `SimIG-Exp` algorithm, and two by the `SimIG-CVaR`. As expected, the `SimIG-Exp` algorithm's solutions have lower values for the *E-RPD* indicator, while the solutions generated using the `SimIG-CVaR` algorithm have an advantage over the *C-RPD* indicator. One can note that all solutions obtained by the `IG` are dominated.

As highlighted, it is up to the decision-makers to choose a solution, according to his risk profile, in this trade-off between risk and expected return. In the example shown, the percentage differences are small. However, depending on the problem, taking a 1% higher risk can significantly impact the company's expected returns, as in the case of oil and gas companies. The operational cost is elevated in offshore oil and gas exploration, and choosing good solutions is even more critical. However, extra statistical information over the solutions can help the decision-makers in this process. As mentioned earlier, any statistic can be evaluated regarding the distribution of the stochastic total

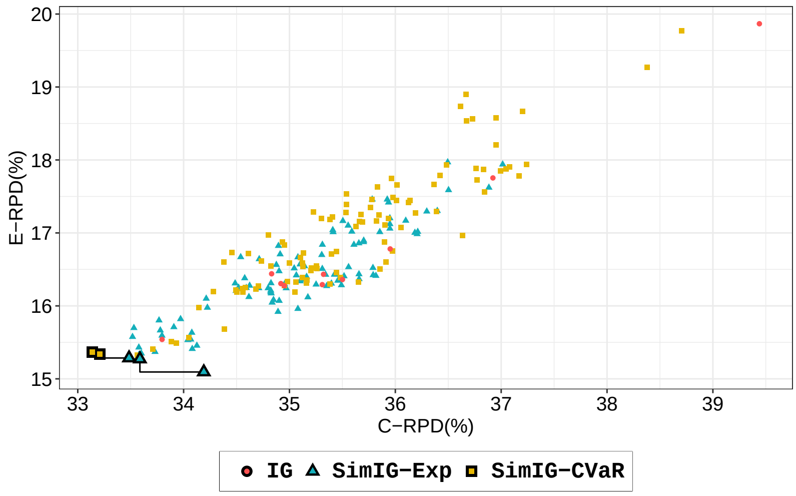
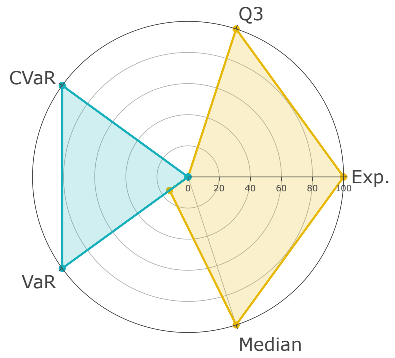


Figure 8.7: Trade-off analysis, between the *E-RPD* and *C-RPD*, of all proposed solutions for instance 50-4-222, considering the high variance level, highlighting the dominant solutions.

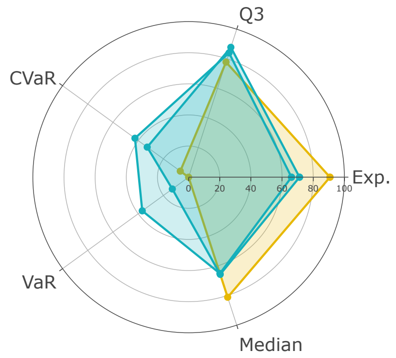
weighted completion times of a solution generated in the simulation step. To reinforce this discussion, we depict in Figure 8.8 the radar plots of the dominant solutions for the selected instance (50-4-222) in terms of the expected objective value (Exp.), the median of the distribution (Median), the third quartile of the distribution (Q3), the value at risk at 95% (VaR), and the conditional value at risk at 95% (CVaR). To allow comparing these statistics in the same scale, we normalize all values within a range between 0 and 100 according to Equation 8-4, where $x = (x_1, ..., x_n)$ is the sample of a specific indicator (Exp., Median, Q3, VaR or CVaR) among the different instances, and z_i is the normalized value of the i^{th} element in x. Figure 8.8a shows these indicators for the two extreme dominant solutions, i.e., the best solution regarding the expected objective value and the best solution regarding the CVaR at 95%. Figure 8.8b depicts the radar plot of the remaining dominant solutions.

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)} \times 100. \tag{8-4}$$

One can note that the best solution regarding the expected objective value presents extreme values for the value at risk and conditional value at risk. Contrariwise, the best solution regarding the CVaR has extreme values for other statistics (Q3 and Median). Thus, depending on the problem, choosing a more balanced solution might be a better decision. Note that the remaining dominant solutions are more balanced regarding the selected indicators and might be a good option depending on the decision-maker, reinforcing their role in the process.



(a) Two extreme dominant solutions.



(b) Remaining dominant solutions.

Figure 8.8: Radar plots of dominant solutions for instance 50-4-222 with different statistics.

8.2.4 Simulation analysis

As previously highlighted, one of the main contributions of our simheuristics is the use of the strategy of building confidence intervals around the mean during the simulation stage. To show the behavior of this procedure within a realistic instance, we present in Figure 8.9 the evolution of the expected objective value during the long simulation step for the 50-4-222 instance. The figure includes the error computed from the confidence intervals, also displayed in the graph. The horizontal axis is shown on a log10 scale for better visualization.

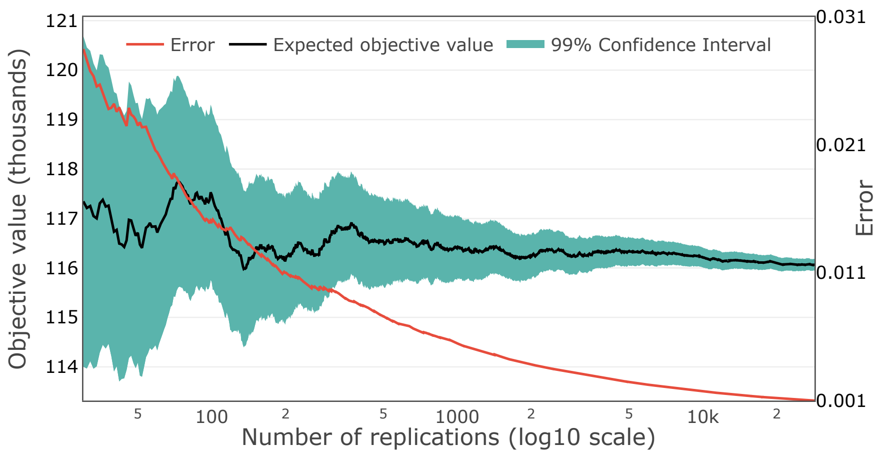


Figure 8.9: Expected objective value evolution for instance 50-4-222 during the long simulation step with the 99% confidence interval around the expected value and with the interval error.

One can notice a higher variation in the expected objective value before 500 replications. After this point, the oscillation decreases while the expected value converges to reach the desired interval error of 0.1%. Note that the simulation requires a large number of replications to achieve the defined error.

It is worth mentioning that, depending on the instance and the problem addressed, the number of replications may vary significantly, emphasizing the importance of the proposed strategy, thus having a more accurate statistic on the simulated sample independently of the problem considered.

8.3 Discussion

In this chapter, we introduced a simheuristic algorithm to solve the PLSVSP with stochastic processing times and release dates for the operations, combining an Iterated Greedy metaheuristic approach with a simulation step to identify promising stochastic solutions. We conducted experiments on the benchmark set of 72 PLSVSP instances, using Log-Normal distributions to model the stochastic parameters. The results show a clear advantage of using the simulation step within the metaheuristic concepts, with statistically significant reductions in the solutions' expected objective value. The computational time overhead is low for the simulation step, resulting in a useful tool for helping the decision-makers during the schedule developments. Moreover, since it generates a pool of stochastic solutions, it allows the decision-makers to choose the schedule that fits their risk profile.

Simheuristics have an important role due to their simplicity and low computational overhead time compared to regular metaheuristics, emerging as an exciting approach for solving stochastic combinatorial optimization problems. The study highlights the importance of using simulation-optimization methods for solving stochastic problems, even when motivated by problems with realistic backgrounds. A risk assessment is presented at the end of the computational experiments section. In this analysis, a simheuristic variant that focuses on minimizing the $\text{CVaR}_{95\%}$ is compared with the approach that aims to minimize the expected value of the solutions' objective function. The comparison makes it possible to identify a set of dominant solutions for a given instance, indicating the Pareto frontier, considering the trade-off between the solutions' expected objective values and the $\text{CVaR}_{95\%}$.

9
Conclusion

The growth of oil and gas exploration in the Brazilian offshore basin, due to the discovery of pre-salt fields, has forced companies operating in the region to pay more attention to the efficient use of their resources. Among these resources, the vessel's fleet is fundamental to their operation, being responsible for tasks extending from the field's development to the production phase. These vessels work in the appraisal, drilling, product distribution, platform support, wells maintenance, among other tasks. This work focused on a specific vessel, specially designed to connect pipelines between subsea oil wells and production platforms in ultra-deepwater regions – The Pipe Laying Support Vessel (PLSV). These vessels are among those with the highest operating and acquisition costs. They not only connect the pipelines but transport them to the wells site after loading it onto their deck. The connection of the wells is the last performed step so that a well can start producing, thus being a task of high impact on the expected annual production of an offshore oil and gas company. Therefore, the PLSV scheduling problem (PLSVSP) consists of servicing a demand of sub-sea oil wells connections, finding the best schedule for a limited PLSV fleet, prioritizing the completion of wells with higher production levels. The problem can be seen as a variant of a batch scheduling problem with parallel machines to minimize the total weighted completion time. In this analogy, vessels are machines, wells represent jobs that must be completed, and batches are the voyages made by the vessels. Each voyage consists of the pipeline loading process in the port, followed by a set of connections to be performed in different wells, regarding the loaded pipelines.

This work addressed a PLSVSP, in its deterministic and stochastic variants, from a company that operates in the Brazilian pre-salt region. Thus, the work had the following objectives: (1) Define the problem properly, modeling it according to some classic formulations of the scheduling literature; (2) Propose a set of benchmark instances for the problem generated from real data of the studied company; (3) Develop heuristic procedures with and without hybridization aiming at improving solutions in terms of quality and computational cost; (4) Propose a stochastic variant of the problem, using an optimization-simulation method to generate solutions that better adjust to uncertainties.

In the first part of the work, we considered only the deterministic variant of the problem, and four mathematical formulations were presented. Two of them were used in two matheuristics, developed to provide faster solutions for the PLSVSP. In the last part regarding the deterministic PLSVSP, we introduce a metaheuristic that uses destruction and repair operators to escape from local optimal solutions and provides faster solutions when compared to the matheuristics. It is worth mentioning that methods that use mathematical formulations, as in the matheuristics, tend to be more flexible in dealing with the studied company's real problem. This is due to the company's management guidelines, which can quickly change the characteristics of the problem. These methods can be easily modified without much computational development effort. We can add or remove constraints in the formulation without the need for any modifications to the matheuristics. A benchmark with 72 instances was generated with different characteristics of the problem to test the methods and formulations. Within the experiments, we observed that the matheuristics were able to improve the solutions generated by pure MIP formulations in less than 1 minute on average, considering the complete set of instances. The average computation time is below 10 minutes for matheuristics, and more than 19,000 seconds for the MIP formulations, also considering the complete set of instances. Finally, we compared the matheuristics with a metaheuristic, with better performance for the metaheuristic in terms of solution quality and computational time even with a high number of iterations. In our experiments, we ran the metaheuristic with a maximum number of 7000 iterations. It was able to generate solutions with an average deviation of 0.07% from the best solutions achieved by the matheuristics in less than 23 seconds on average. In comparison, the best matheuristic's average deviation is 0.74%. However, as highlighted previously, modifications in the metaheuristic requires a more significant computational development effort. Providing different optimization methods allows the company to evaluate them comparatively and define the one that better adapts to their process.

In the second part of the work, we consider a stochastic variant of the PLSVSP in which some of the problem parameters are random variables, following non-negative probability distributions. PLSVSP's main uncertainties regard the pipeline connections' duration and pipelines' arrival dates at the port. The consideration of uncertainties represents a more practical problem, making it possible to develop better solutions that suit the realistic environment. From our knowledge, the present work is the first one dealing with the stochastic PLSVSP. We developed a simheuristic technique, defined by combining a metaheuristic structure with embedded Monte Carlo simulation, to solve the problem. These methods have shown excellent results in dealing with stochastic scheduling problems in the literature. We tested the simheuristic within different variability scenarios for the uncertain parameters. Results showed an advantage in using built-in simulation to deal with the stochastic PLSVSP, with significantly better solutions in terms of expected costs compared to a deterministic metaheuristic, without much computational time overhead. The method provides a pool of stochastic solutions, allowing the decision-makers to choose the schedule that fits their risk profile. Thus, we present a Pareto frontier analysis, considering the trade-off between the solutions' expected objective values and the $\text{CVaR}_{95\%}$.

We fulfill all the thesis's objectives by introducing several optimization methods to deal with the PLSVSP in its deterministic and stochastic variants. To the best of our knowledge, our work is the first to formulate and deal with a complete PLSVSP version. Unlike other strategies in the literature, our approaches allow the problem to be solved without reducing the solution space, diminishing the risk of removing the optimal or high-quality solutions.

We limited our study to the PLSVSP in the Brazilian pre-salt region. Other companies or regions may define different rules on how to schedule the PLSV fleet. Some consider precedence constraints for the pipeline connection tasks, others aim to minimize the tasks' tardiness according to their pre-established due dates, others deal with smaller capacity vessels, among other characteristics. We also disregarded the occurrence of disruptions within the schedule, such as ship wreckage, ship maintenance, equipment failure, and others. The practical problem is susceptible to operational issues that can interrupt one or more services in the wells. In the stochastic variant of the problem, we limited the problem to two uncertainties. However, planners might consider other relevant uncertainties to make the simulation step even more suitable to the real process, such as duration uncertainty varying according to the vessel, uncertainty on the pipeline loading times, and vessel eligibility.

A significant contribution of our work is to approach a complex real-life scheduling problem with several aspects and constraints. Moreover, based on the relations drawn between the studied problem and a parallel machine scheduling problem, the work collaborates with this crucial area approached by the operations research community. The results obtained show that the planners can use the developed tools in practice and that, since it is a complex machine scheduling problem, it can assist in similar works or simplified variants of the PLSVSP, such as problems with batch scheduling, open shop scheduling problems, family scheduling problems, among others. The PLSVSP combines several machine scheduling aspects simultaneously, such as job splitting, machine eligibility, release dates, non-anticipatory setup times, and others. Most of the works found in the machine scheduling literature do not consider more than two or three aspects. Furthermore, one of the most challenging considered aspects, the non-anticipatory family setup times, is rarely found in the scheduling literature. To the best of our knowledge, our work is the first to model the non-anticipatory setup times within a family scheduling problem. Despite being a widely studied field, the machine scheduling theory encompasses several industrial problems with their specificities, giving good perspectives for new studies. We believe that family and batch scheduling problems are of interest to many industries and deserve sustained attention. Researchers could investigate some interesting variants as unrelated parallel machine scheduling problems with non-anticipatory and machine-dependent family setup times, machine scheduling problems with sequence-dependent non-anticipatory family setup times, and serial batch scheduling with family setup times and job availability. In batch scheduling problems with job availability, a task concludes according to its completion time instead of its assigned batch completion time.

9.1 Future Perspectives

Below we list some research perspectives for the continuity of the present work:

- Study the PLSVSP regarding other exploratory regions in Brazil. It is worth mentioning that given the similarity between the problems, it is expected that few modifications in the methods are necessary.
- Consider disruptions to the problem, such as ship wreckage, ship maintenance, equipment failure, and others. These interruptions can be studied and considered *a priori* in a deterministic or stochastic approach based on historical data.
- Study the integrated problem in which the PLSV fleet is scheduled in conjunction with rigs to optimize the development phase of the wells. Currently, the company defines the rigs' schedules before the PLSV fleet further. However, integrating these problems can generate an interesting research study and result in financial and operational gains for the company.
- Implement the methods proposed in the studied company to test it in the real workday of the planners.

- Evaluate other optimization approaches, as bio-inspired methods like genetic algorithms, memetic algorithms, particle swarm optimization algorithms, and others.
- Evaluate the performance of the methods in generalized instances. All experiments were carried out on instances generated with real data from the PLSVSP in the Brazilian pre-salt region. However, we can use the machine scheduling literature to generate a more general set of instances, following the most usual rules and distributions.
- Test the methods within established machine scheduling instances. The problem's complexity allows it to be simplified, resulting in other known machine scheduling problems. Thus, it would be interesting to evaluate the performance of the developed methods in solving these problems.
- Develop other stochastic optimization approaches for solving the PLSVSP.
- Consider other probability distributions for modeling the stochastic parameters.
- Approach the problem as an unrelated parallel machine scheduling problem, in which the duration of tasks varies according to the assigned machine to perform it. Also, extend the problem for the stochastic variant of it, in which the uncertainty on the duration also depends on the assigned machine.
- Assess the wait-and-see solutions in the stochastic PLSVSP. In this analysis, the metaheuristic must be performed in each stochastic scenario to generate the expected average value of the objective function if the uncertainty parameters were known before the optimization. This analysis can help to assess the impact of uncertainties in the problem.
- Test different statistics of the input distributions for the stochastic parameter during the deterministic evaluation of solution in the simheuristic, instead of only considering the distribution's mean.
- Consider the convex combination between the expected value and the CVaR as the statistic selection for the simheuristic, allowing it to find balanced solution regarding these aspects during its execution.
- Extend the simheuristic to a multi-objective approach, in which a weight parameter defines which objective to prioritize. The idea is to adjust the weight iteratively to build a complete set of Pareto Optimal solutions. One objective may be the expected objective value in this approach, and the other may be the CVaR.

– Extend the simheuristic to a multi-objective approach, in which a weight parameter defines which objective to prioritize. The idea is to adjust the weight iteratively to build a complete set of Pareto Optimal solutions. One objective may be the expected objective value in this approach, and the other may be the CVaR.

Bibliography

Abu-Marrul, V., Martinelli, R., and Hamacher, S. (2019). Instances for the plsv scheduling problem: An identical parallel machine approach with non-anticipatory family setup times. URL: https://doi.org/10.17771/PUCRio.ResearchData.45799.

Abu-Marrul, V., Martinelli, R., and Hamacher, S. (2020). Scheduling pipe laying support vessels with non-anticipatory family setup times and intersections between sets of operations. *International Journal of Production Research*, 0(0):1–15.

Abu-Marrul, V., Martinelli, R., Hamacher, S., and Gribkovskaia, I. (2021a). Matheuristics for a parallel machine scheduling problem with non-anticipatory family setup times: Application in the offshore oil and gas industry. *Computers & Operations Research*.

Abu-Marrul, V., Mecler, D., Martinelli, R., Hamacher, S., and Gribkovskaia, I. (2021b). Heuristics for Scheduling Pipe-laying Support Vessels: An Identical Parallel Machine Scheduling Approach. In 17^{th} *International Workshop on Project Management and Scheduling*.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246(2):345–378.

Allahyarzadeh-Bidgoli, A., Salviano, L. O., Dezan, D. J., de Oliveira Junior, S., and Yanagihara, J. I. (2018). Energy optimization of an fpso operating in the brazilian pre-salt region. *Energy*, 164:390–399.

Azizoglu, M. and Webster, S. (2003). Scheduling parallel machines to minimize weighted flowtime with family set-up times. *International Journal of Production Research*, 41(6):1199–1215.

Bassi, H. V., Ferreira Filho, V. J. M., and Bahiense, L. (2012). Planning and scheduling a fleet of rigs using simulation–optimization. *Computers & Industrial Engineering*, 63(4):1074–1088.

Beck, J. C., Prosser, P., and Selensky, E. (2002). On the reformulation of vehicle routing problems and scheduling problems. In *International symposium on abstraction, reformulation, and approximation*, pages 282–289. Springer.

Beck, J. C., Prosser, P., and Selensky, E. (2003). Vehicle routing and job shop scheduling: What's the difference? In *ICAPS*, pages 267–276.

Behera, D. K. (2012). Complexity on parallel machine scheduling: A review. In *Emerging Trends in Science, Engineering and Technology*, pages 373–381. Springer.

Beltrao, R. L. C., Sombra, C. L., Lage, A. C. V., Netto, J. R. F., Henriques, C. C. D., et al. (2009). Pre-salt santos basin-challenges and new technologies for the development of the pre-salt cluster, santos basin, brazil (otc-19880). In *Offshore Technology Conference*, Houston, Texas, USA.

Bettayeb, B., Kacem, I., and Adjallah, K. H. (2008). An improved branch-and-bound algorithm to minimize the weighted flowtime on identical parallel machines with family setup times. *Journal of Systems Science and Systems Engineering*, 17(4):446–459.

Billaut, J.-C., Della Croce, F., and Grosso, A. (2015). A single machine scheduling problem with two-dimensional vector packing constraints. *European Journal of Operational Research*, 243(1):75–81.

Calvet, L., Wang, D., Juan, A., and Bové, L. (2019). Solving the multidepot vehicle routing problem with limited depot capacity and stochastic demands. *International Transactions in Operational Research*, 26(2):458–484.

Chen, Z.-L. and Powell, W. B. (2003). Exact algorithms for scheduling multiple families of jobs on parallel machines. *Naval Research Logistics*, 50(7):823–840.

Cheng, T. and Sin, C. (1990). A state-of-the-art review of parallel-machine scheduling research. *European Journal of Operational Research*, 47(3):271–292.

Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2007). Maritime transportation. *Handbooks in operations research and management science*, 14:189–284.

Christiansen, M., Fagerholt, K., Nygreen, B., and Ronen, D. (2013). Ship routing and scheduling in the new millennium. *European Journal of Operational Research*, 228(3):467–483.

Christiansen, M., Fagerholt, K., and Ronen, D. (2004). Ship routing and scheduling: Status and perspectives. *Transportation science*, 38(1):1–18.

Ciavotta, M., Meloni, C., and Pranzo, M. (2016). Speeding up a Rollout algorithm for complex parallel machine scheduling. *International Journal of Production Research*, 54(16):4993–5009.

Clevelario, J., Pires, F., Barros, C., and Sheldrake, T. (2010). Flexible pipe systems configurations for the pre-salt area. In *ASME 2010 29th International Conference on Ocean, Offshore and Arctic Engineering*, pages 457–464. American Society of Mechanical Engineers Digital Collection.

Cunha, V., Santos, I., Pessoa, L., and Hamacher, S. (2020). An ILS heuristic for the ship scheduling problem: application in the oil industry. *International Transactions in Operational Research*, 27(1):197–218.

De Lima, H. F. (2007). Metodologia para a tomada de decisão no projeto de sistemas submarinos de produção de óleo e gás. 2007. 169 p. Master's thesis, COPPE/UFRJ. Rio de Janeiro.

Della Croce, F., Grosso, A., and Salassa, F. (2014). A matheuristic approach for the two-machine total completion time flow shop problem. *Annals of Operations Research*, 213(1):67–78.

Della Croce, F., Grosso, A., and Salassa, F. (2019). Minimizing total completion time in the two-machine no-idle no-wait flow shop problem. *Journal of Heuristics*, pages 1–15.

Devold, H. (2013). *Oil and gas production handbook: an introduction to oil and gas production, transport, refining and petrochemical industry*. ABB Oil and Gas.

Dunstall, S. and Wirth, A. (2005a). A comparison of branch-and-bound algorithms for a family scheduling problem with identical parallel machines. *European Journal of Operational Research*, 167(2):283–296.

Dunstall, S. and Wirth, A. (2005b). Heuristic methods for the identical parallel machine flowtime problem with set-up times. *Computers & Operations Research*, 32(9):2479–2491.

Edis, E. B., Oguz, C., and Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, 230(3):449–463.

Ekici, A., Elyasi, M., Özener, O. Ö., and Sarıkaya, M. B. (2019). An application of unrelated parallel machine scheduling with sequence-dependent setups at vestel electronics. *Computers & Operations Research*, 111:130–140.

Eom, D.-H., Shin, H.-J., Kwun, I.-H., Shim, J.-K., and Kim, S.-S. (2002). Scheduling Jobs on Parallel Machines with Sequence-Dependent Family Set-up Times. *The International Journal of Advanced Manufacturing Technology*, 19(12):926–932.

Fabri, M. and Ramalhinho, H. (2021). The in-house logistics routing problem. *International Transactions in Operational Research*.

Fanjul-Peyro, L., Perea, F., and Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. *European Journal of Operational Research*, 260(2):482–493.

Fanjul-Peyro, L. and Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1):55–69.

Fernández Pérez, M. A., Oliveira, F., and Hamacher, S. (2018). Optimizing workover rig fleet sizing and scheduling using deterministic and stochastic programming models. *Industrial & Engineering Chemistry Research*, 57(22):7544–7554.

Fonseca-Reyna, Y. C., Martínez-Jiménez, Y., Cabrera, A. V., and Sanchez, E. A. R. (2019). Optimization of heavly constrained hybrid-flexible slowshop problems using a multi-agent reinforcement learning approach. *Investigación Operacional*, 40(1):100–111.

Fuchigami, H. Y., Moccellin, J. V., and Ruiz, R. (2015). New priority rules for the flexible flow line scheduling problem with setup times. *Production*, 25(4):779–790.

Gokhale, R. and Mathirajan, M. (2012). Scheduling identical parallel machines with machine eligibility restrictions to minimize total weighted flowtime in automobile gear manufacturing. *The International Journal of Advanced Manufacturing Technology*, 60(9-12):1099–1110.

González, M. A., Vela, C. R., Varela, R., and González-Rodríguez, I. (2015). An advanced scatter search algorithm for solving job shops with sequence dependent and non-anticipatory setups. *AI Communications*, 28(2):179–193.

Gonzalez-Martin, S., Juan, A. A., Riera, D., Elizondo, M. G., and Ramos, J. J. (2018). A simheuristic algorithm for solving the arc routing problem with stochastic demands. *Journal of Simulation*, 12(1):53–66.

Gonzalez-Neira, E. M., Ferone, D., Hatami, S., and Juan, A. A. (2017). A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 79:23–36.

González-Neira, E. M., Urrego-Torres, A. M., Cruz-Riveros, A. M., Henao-García, C., Montoya-Torres, J. R., Molina-Sánchez, L. P., and Jiménez, J.-F. (2019). Robust solutions in multi-objective stochastic permutation flow shop problem. *Computers & Industrial Engineering*, 137:106026.

Grasas, A., Juan, A. A., and Lourenço, H. R. (2016). Simils: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, 10(1):69–77.

Grenouilleau, F., Lahrichi, N., and Rousseau, L.-M. (2020). New decomposition methods for home care scheduling with predefined visits. *Computers & Operations Research*, 115:104855.

Gruler, A., Panadero, J., de Armas, J., Pérez, J. A. M., and Juan, A. A. (2018). Combining variable neighborhood search with simulation for the inventory routing problem with stochastic demands and stock-outs. *Computers & Industrial Engineering*, 123:278–288.

Gruler, A., Panadero, J., de Armas, J., Pérez, J. A. M., and Juan, A. A. (2020). A variable neighborhood search simheuristic for the multiperiod inventory routing problem with stochastic demands. *International Transactions in Operational Research*, 27(1):314–335.

Gruler, A., Quintero-Araújo, C. L., Calvet, L., and Juan, A. A. (2017). Waste collection under uncertainty: a simheuristic based on variable neighbourhood search. *European Journal of Industrial Engineering*, 11(2):228–255.

Guimarans, D., Dominguez, O., Panadero, J., and Juan, A. A. (2018). A simheuristic approach for the two-dimensional vehicle routing problem with stochastic travel times. *Simulation Modelling Practice and Theory*, 89:1–14.

Haddad, E. and Giuberti, A. C. (2010). Economic impacts of pre-salt on a regional economy: the case of espírito santo, brazil. In *50th Congress of the European Regional Science Association:"Sustainable Regional Growth and Development in* *the Creative Knowledge Economy"*, Jönköping, Sweden. European Regional Science Association (ERSA).

Ham, A., Fowler, J. W., and Cakici, E. (2017). Constraint programming approach for scheduling jobs with release times, non-identical sizes, and incompatible families on parallel batching machines. *IEEE Transactions on Semiconductor Manufacturing*, 30(4):500–507.

Hansen, P. and Mladenović, N. (2003). Variable neighborhood search. In *Handbook of metaheuristics*, pages 145–184. Springer.

Hatami, S., Calvet, L., Fernández-Viagas, V., Framiñán, J. M., and Juan, A. A. (2018). A simheuristic algorithm to set up starting times in the stochastic parallel flowshop problem. *Simulation Modelling Practice and Theory*, 86:55–71.

Heath, S. K., Bard, J. F., and Morrice, D. J. (2013). A grasp for simultaneously assigning and sequencing product families on flexible assembly lines. *Annals of Operations Research*, 203(1):295–323.

Islam, M. and Khan, M. (2013). *The petroleum engineering handbook: sustainable operations*. Elsevier.

Juan, A., Faulin, J., Grasman, S., Riera, D., Marull, J., and Mendez, C. (2011). Using safety stocks and simulation to solve the vehicle routing problem with stochastic demands. *Transportation Research Part C: Emerging Technologies*, 19(5):751–765.

Juan, A. A., Barrios, B. B., Vallada, E., Riera, D., and Jorba, J. (2014). A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. *Simulation Modelling Practice and Theory*, 46:101–117.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., and Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, 2:62–72.

Kaabi, J. and Harrath, Y. (2014). A survey of parallel machine scheduling under availability constraints. *International Journal of Computer and Information Technology*, 3(2):238–245.

Kalinowski, T., Matthews, J., and Waterer, H. (2020). Scheduling of maintenance windows in a mining supply chain rail network. *Computers & Operations Research*, 115:104670.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Kouki, Z., Chaar, B. F., Hammadi, S., and Ksouri, M. (2007). Analogies between flexible job shop scheduling and vehicle routing problems. In *2007 IEEE International Conference on Industrial Engineering and Engineering Management*, pages 880–884. IEEE.

LABANCA, E. L. (2005). Metodologia para a seleção de arranjos submarinos baseada na eficiência operacional. Master's thesis, Universidade Federal do Rio de Janeiro.

Lam, K. and Xing, W. (1997). New trends in parallel machine scheduling. *International Journal of Operations & Production Management*.

Latorre-Biel, J. I., Ferone, D., Juan, A. A., and Faulin, J. (2021). Combining simheuristics with petri nets for solving the stochastic vehicle routing problem with correlated demands. *Expert Systems with Applications*, 168:114240.

Lawrence, S. A. (1972). *International sea transport: the years ahead*. Lexington Books.

Lee, C. (2017). A dispatching rule and a random iterated greedy metaheuristic for identical parallel machine scheduling to minimize total tardiness. *International Journal of Production Research*, 56:1–17.

Lee, Y. H. and Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. *European Journal of Operational Research*, 100(3):464–474.

Li, K. and Yang, S.-l. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. *Applied mathematical modelling*, 33(4):2145–2158.

Liao, C.-J., Chao, C.-W., and Chen, L.-C. (2012). An improved heuristic for parallel machine weighted flowtime scheduling with family set-up times. *Computers & Mathematics with Applications*, 63(1):110–117.

Lin, B. M. and Cheng, T. E. (2005). Two-machine flowshop batching and scheduling. *Annals of Operations Research*, 133(1-4):149–161.

Lin, S.-W. and Ying, K.-C. (2016). Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics. *Omega*, 64:115–125.

Lin, S.-W. and Ying, K.-C. (2019). Makespan optimization in a no-wait flowline manufacturing cell with sequence-dependent family setup times. *Computers & Industrial Engineering*, 128:1–7.

Lopes, T. C., Michels, A. S., Lüders, R., and Magatão, L. (2020). A simheuristic approach for throughput maximization of asynchronous buffered stochastic mixed-model assembly lines. *Computers & Operations Research*, 115:104863.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2003). Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer.

Martinelli, R., Collard, J., and Gamache, M. (2019). Strategic planning of an underground mine with variable cut-off grades. *Optimization and Engineering*, pages 1–47.

Mecler, D. (2020). A metaheuristic for the pipe laying support vessel scheduling problem. Master's thesis, Pontifical Catholic University of Rio de Janeiro - PUC-Rio.

Mecler, D., Abu-Marrul, V., Martinelli, R., and Hoff, A. (2021). Iterated greedy algorithms for a complex parallel machine scheduling problem. *arXiv*.

Mehdizadeh, E., Tavakkoli-Moghaddam, R., and Yazdani, M. (2015). A vibration damping optimization algorithm for a parallel machines scheduling problem with sequence-independent family setup times. *Applied Mathematical Modelling*, 39(22):6845–6859.

Mendes, A. B. (2007). *Programação de frota de apoio a operações\'offshore\'sujeita à requisição de múltiplas embarcações para uma mesma tarefa.* PhD thesis, Universidade de São Paulo.

Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. *Asia-Pacific Journal of Operational Research*, 18(2):193.

Mönch, L. and Roob, S. (2018). A matheuristic framework for batch machine scheduling problems with incompatible job families and regular sum objective. *Applied Soft Computing*, 68:835–846.

Monemi, R. N., Danach, K., Khalil, W., Gelareh, S., Lima Jr, F. C., and Aloise, D. J. (2015). Solution methods for scheduling of heterogeneous parallel machines applied to the workover rig problem. *Expert Systems with Applications*, 42(9):4493–4505.

National Oilwell Varco (2020). Flexible pipe system accessories.

O Petróleo (2017). Sapura energy ganha us$ 352 milhões em contratos na malásia e no brasil.

Obeid, A., Dauzère-Pérès, S., and Yugma, C. (2014). Scheduling job families on non-identical parallel machines with time constraints. *Annals of Operations Research*, 213(1):221–234.

Offshore Energy Today (2013). Subsea 7 scores $600 mln in petrobras plsv contracts.

Omar, M. K. and Teo, S. C. (2006). Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach. *Applied Mathematics and Computation*, 181(2):1008–1017.

Onggo, B. S., Panadero, J., Corlu, C. G., and Juan, A. A. (2019). Agri-food supply chains with stochastic demands: A multi-period inventory routing problem with perishable products. *Simulation Modelling Practice and Theory*, 97:101970.

Ozer, E. A. and Sarac, T. (2019). Mip models and a matheuristic algorithm for an identical parallel machine scheduling problem under multiple copies of shared resources constraints. *Top*, 27(1):94–124.

Pagès-Bernaus, A., Ramalhinho, H., Juan, A. A., and Calvet, L. (2019). Designing e-commerce supply chains: a stochastic facility–location approach. *International Transactions in Operational Research*, 26(2):507–528.

Panadero, J., Doering, J., Kizys, R., Juan, A. A., and Fito, A. (2020). A variable neighborhood search simheuristic for project portfolio selection under uncertainty. *Journal of Heuristics*, 26(3):353–375.

Pinedo, M. (2012). *Scheduling. Theory, algorithms, and systems*, volume 29. Springer.

Pisinger, D. and Røpke, S. (2007). A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435.

Potts, C. N. and Kovalyov, M. Y. (2000). Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249.

Queiroz, M. M. and Mendes, A. B. (2011). Heuristic approach for solving a pipe layer fleet scheduling problem. In Rizzuto, E. and Soares, C. G., editors, *Sustainable Maritime Transportation and Exploitation of Sea Resources*, chapter 9, pages 1073–1080. Taylor & Francis Group, London.

Quintero-Araujo, C. L., Gruler, A., Juan, A. A., de Armas, J., and Ramalhinho, H. (2017). Using simheuristics to promote horizontal collaboration in stochastic city logistics. *Progress in Artificial Intelligence*, 6(4):275–284.

Quintero-Araujo, C. L., Guimarans, D., and Juan, A. A. (2019). A simheuristic algorithm for the capacitated location routing problem with stochastic demands. *Journal of Simulation*, pages 1–18.

Raba, D., Estrada-Moreno, A., Panadero, J., and Juan, A. A. (2020). A reactive simheuristic using online data for a real-life inventory routing problem with stochastic demands. *International Transactions in Operational Research*.

Rabbani, M., Heidari, R., and Yazdanparast, R. (2019). A stochastic multi-period industrial hazardous waste location-routing problem: Integrating nsga-ii and monte carlo simulation. *European Journal of Operational Research*, 272(3):945–961.

Resende, M. G. C. and Ribeiro, C. C. (2019). *Greedy Randomized Adaptive Search Procedures: Advances and Extensions*, pages 169–220. Springer International Publishing, Cham.

Reyes-Rubiano, L., Ferone, D., Juan, A. A., and Faulin, J. (2019). A simheuristic for routing electric vehicles with limited driving ranges and stochastic travel times. *SORT-Statistics and Operations Research Transactions*, 1(1):3–24.

Rodrigues, L. A. and Sauer, I. L. (2015). Exploratory assessment of the economic gains of a pre-salt oil field in brazil. *Energy Policy*, 87:486–495.

Rodriguez, F. J., Blum, C., García-Martínez, C., and Lozano, M. (2012). Grasp with path-relinking for the non-identical parallel machine scheduling problem with minimising total weighted completion times. *Annals of Operations Research*, 201(1):383–401.

Ronen, D. (1983). Cargo ships routing and scheduling: Survey of models and problems. *European Journal of Operational Research*, 12(2):119–126.

Ronen, D. (1993). Ship scheduling: The last decade. *European Journal of Operational Research*, 71(3):325–333.

Ropke, S. and Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472.

Roshanaei, V., Balagh, A. K. G., Esfahani, M. M. S., and Vahdani, B. (2010). A mixed-integer linear programming model along with an electromagnetism-like algorithm for scheduling job shop production system with sequence-dependent set-up times. *The International Journal of Advanced Manufacturing Technology*, 47(5-8):783–793.

Ruiz, R., Pan, Q.-K., and Bahman, N. (2019). Iterated greedy methods for the distributed permutation flowshop scheduling problem. *Omega*, 83:213–222.

Ruiz, R., Şerifoğlu, F. S., and Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4):1151–1175.

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

Santos, M. S., Pinto, T. V., Ênio Lopes Júnior, Cota, L. P., Souza, M. J., and Euzébio, T. A. (2020). Simheuristic-based decision support system for efficiency improvement of an iron ore crusher circuit. *Engineering Applications of Artificial Intelligence*, 94:103789.

Schaller, J. E. (2014). Minimizing total tardiness for scheduling identical parallel machines with family setups. *Computers & Industrial Engineering*, 72:274–281.

Shin, H. J. and Leon, V. J. (2004). Scheduling with product family set-up times: an application in TFT LCD manufacturing. *International Journal of Production Research*, 42(20):4235–4248.

SINAVAL (2013). Technip e dof conquistam contrato para quatro navios plsv.

Speight, J. G. (2015). *Handbook of Offshore Oil and Gas Operations*. Elsevier.

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

Su, L.-H. (2009). Scheduling on identical parallel machines to minimize total completion time with deadline and machine eligibility constraints. *The International Journal of Advanced Manufacturing Technology*, 40(5-6):572–581.

Subramanian, A., Drummond, L. M. d. A., Bentes, C., Ochi, L. S., and Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 37(11):1899–1911.

Ta, Q. C., Billaut, J.-C., and Bouquard, J.-L. (2018). Matheuristic algorithms for minimizing total tardiness in the m-machine flow-shop scheduling problem. *Journal of Intelligent Manufacturing*, 29(3):617–628.

Tavakkoli-Moghaddam, R., Tavakkoli-Moghaddam, R., and Mehdizadeh, E. (2007). A new ilp model for identical parallel-machine scheduling with family setup times minimizing the total weighted flow time by a genetic algorithm. Technical Report 2.

TecPetro (2015). Dutos submarinos.

Thomas, J. E. (2001). *Fundamentos de engenharia de petróleo*. Interciência.

Thompson, J. M. (2018). Exact or metaheuristic methods or a bit of both–the rise of matheuristics. *Keynote Papers*, page 32.

Tseng, C.-T. and Lee, C.-H. (2017). A new electromagnetism-like mechanism for identical parallel machine scheduling with family setup times. *The International Journal of Advanced Manufacturing Technology*, 89(5-8):1865–1874.

UNCTAD, U. (2015). World investment report 2015: Reforming international investment governance. *United Nations Publications Customer Service*, page 253.

Unlu, Y. and Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers & Industrial Engineering*, 58(4):785–800.

Đurasević, M. and Jakobović, D. (2018). A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications*, 113:555–569.

Van Der Zee, D.-J. (2015). Family-based dispatching with parallel machines. *International Journal of Production Research*, 53:5837–5856.

Villarinho, P. A., Panadero, J., Pessoa, L. S., Juan, A. A., and Oliveira, F. L. C. (2021). A simheuristic algorithm for the stochastic permutation flow-shop problem with delivery dates and cumulative payoffs. *International Transactions in Operational Research*, 28(2):716–737.

Webster, S. and Azizoglu, M. (2001). Dynamic programming algorithms for scheduling parallel machines with family setup times. *Computers & Operations Research*, 28(2):127–137.

Weng, M. X., Lu, J., and Ren, H. (2001). Unrelated parallel machine scheduling with setup consideration and a total weighted completion time objective. *International journal of production economics*, 70(3):215–226.

Woo, Y.-B. and Kim, B. S. (2018). Matheuristic approaches for parallel machine scheduling problem with time-dependent deterioration and multiple rate-modifying activities. *Computers & Operations Research*, 95:97–112.

Yazdani, M., Kabirifar, K., Frimpong, B. E., Shariati, M., Mirmozaffari, M., and Boskabadi, A. (2021). Improving construction and demolition waste collection service in an urban area using a simheuristic approach: A case study in sydney, australia. *Journal of Cleaner Production*, 280:124138.

A
Publications

Our publication strategy followed the order in which the thesis introduces the optimization approaches. In this way, each chapter of methods (Chapters 4 to 8) generated a research paper, with four full research articles and one extended abstract. The extended abstract is available in the proceedings of a conference specialized in project management and scheduling problems. Among the full articles, two were published in leading journals on applied operations research and industrial engineering problems. And, the remaining papers were submitted to high-standard operations research journals with pre-print versions available online. More details about the research papers developed during the thesis are given in the following:

Paper 1

Title:	Scheduling pipe laying support vessels with non-anticipatory family setup times and intersections between sets of operations (Abu-Marrul et al. 2020).
Authors:	Victor Abu-Marrul, Rafael Martinelli, and Silvio Hamacher.
Content:	Mathematical Formulations for the PLSVSP (Chapter 4).
Type:	Full research paper.
Journal or Conference:	International Journal of Production Research (IJPR).
Status:	Published (2020).

Paper 2

Title:	Heuristics for Scheduling Pipe-laying Support Vessels: An Identical Parallel Machine Scheduling Approach (Abu-Marrul et al. 2021b).
Authors:	Victor Abu-Marrul, Davi Mecler, Rafael Martinelli, Silvio Hamacher, and Irina Gribkovskaia.
Content:	Constructive Heuristics for the PLSVSP (Chapter 5).
Type:	Extended abstract.
Journal or Conference:	17^{th} International Workshop on Project Management and Scheduling (PMS).
Status:	In proceedings (2021).

Paper 3

Title:	Matheuristics for a parallel machine scheduling problem with non-anticipatory family setup times: Application in the offshore oil and gas industry (Abu-Marrul et al. 2021a).
Authors:	Victor Abu-Marrul, Rafael Martinelli, Silvio Hamacher, and Irina Gribkovskaia.
Content:	MIP-based Neighborhood Search Matheuristics for the PLSVP (Chapter 6).
Type:	Full research paper.
Journal or Conference:	Computers and Operations Research (COR).
Status:	Published (2021).

Paper 4

Title:	Iterated Greedy Algorithms for a Complex Parallel Machine Scheduling Problem" (Mecler et al. 2021).
Authors:	Davi Mecler, Victor Abu-Marrul, Rafael Martinelli, and Arild Hoff.
Content:	Iterated Greedy Algorithm for the PLSVSP (Chapter 7) and other Iterated Greedy algorithm variants, applied to a new benchmark set of instances.
Type:	Full research paper.
Journal or Conference:	European Journal of Operational Research (EJOR)
Status:	Submitted/Pre-print available (2021).

Paper 5

Title:	Simheuristic algorithm for a stochastic parallel machine scheduling problem.
Authors:	Victor Abu-Marrul, Rafael Martinelli, Silvio Hamacher, and Irina Gribkovskaia.
Content:	Iterated Greedy Simheuristic with Embedded Monte Carlo Simulation for the stochastic PLSVSP (Chapter 8).
Type:	Full research paper.
Journal or Conference:	Annals of Operations Research (ANOR)
Status:	Submitted to a special issue entitled "Recent Advances in Simulation-based Optimization for Operations Research Problems" (2021).

B
Mathematical Formulations

B.1
Tables of Symbols

B.1.1
General Elements

Table B.1: General Elements for all formulations.

Type	Name	Description
Set	$\mathcal{O}$	Set of operations to be scheduled.
Set	$\mathcal{M}$	Set of heterogeneous machines.
Set	$\mathcal{N}$	Set of jobs, which group operations.
Set	$\mathcal{F}$	Set of families.
Set	$\mathcal{O}_j$	Subset of operations composing job $J_j \in \mathcal{N}$.
Set	$\mathcal{O}_g$	Subset of operations that composes to family $F_g \in \mathcal{F}$.
Set	$\mathcal{M}_i$	Subset of machines $M_k \in \mathcal{M}$ eligible to process operation $O_i \in \mathcal{O}$.
Set	$\mathcal{N}_i$	Subset of jobs related to operation $O_i \in \mathcal{O}$.
Parameter	p_i	Processing time of operation $O_i \in \mathcal{O}$.
Parameter	r_i	Release date of operation $O_i \in \mathcal{O}$.
Parameter	l_i	Load occupation of operation $O_i \in \mathcal{O}$.
Parameter	w_j	Weight of job $J_j \in \mathcal{N}$.
Parameter	w_i	Relative weight of operation $O_i \in \mathcal{O}$, defined as $\max\limits_{J_j \in \mathcal{N}_i}\{w_j\}$.
Parameter	r_k	Release date of machine $M_k \in \mathcal{M}$.
Parameter	q_k	Capacity limit of machine $M_k \in \mathcal{M}$ (usually 1.0 or 100).
Parameter	s_g	Setup time of family $F_g \in \mathcal{F}$.
Parameter	r_{max}	Maximum operations release date, defined as $\max\limits_{O_i \in \mathcal{O}}\{r_i\}$.
Parameter	M	Large number.

B.1.2
Specific Formulations Elements

Table B.2: Positional Scheduling Formulation Elements.

Type	Name	Description
Set	$\mathcal{P}_k$	Set of positions on machine $M_k \in \mathcal{M}$.
Binary variable	X_{ik}^p	If operation $O_i \in \mathcal{O}$ is scheduled in the p-th position on machine $M_k \in \mathcal{M}$.
Binary variable	Y_{gk}^p	If a setup time of family $F_g \in \mathcal{F}$ is scheduled in p-th position on machine $M_k \in \mathcal{M}$.
Continuous variable	S_k^p	Starting time of the p-th position on machine $M_k \in \mathcal{M}$.
Continuous variable	L_k^p	Total Load of the p-th position on machine $M_k \in \mathcal{M}$.
Continuous variable	R_k^p	Release of the p-th position on machine $M_k \in \mathcal{M}$.
Continuous variable	C_i	Completion time of operation $O_i \in \mathcal{O}$.
Continuous variable	C_j	Completion time of job $J_j \in \mathcal{N}$.

Table B.3: Time-Index Scheduling Formulation Elements.

Type	Name	Description
Set	$\mathcal{T}_k$	Set of periods on machine $M_k \in \mathcal{M}$.
Binary variable	X_{ik}^t	If operation $O_i \in \mathcal{O}$ is start on period $t \in \mathcal{T}$ on machine $M_k \in \mathcal{M}$.
Binary variable	Y_{gk}^t	If the a setup of family $F_g \in \mathcal{F}$ starts at period $t \in \mathcal{T}$ on machine $M_k \in \mathcal{M}$.
Continuous variable	L_k^t	Total accumulated load at period $t \in \mathcal{T}$ on machine $M_k \in \mathcal{M}$.
Continuous variable	R_k^t	Release defined for period $t \in \mathcal{T}$ on machine $M_k \in \mathcal{M}$.
Continuous variable	C_i	Completion time of operation $O_i \in \mathcal{O}$.
Continuous variable	C_j	Completion time of job $J_j \in \mathcal{N}$.

Table B.4: Batch Scheduling Formulations Elements.

Type	Name	Description
Set	$\mathcal{B}_k$	Set of batches on machine $M_k \in \mathcal{M}$.
Set	$\mathcal{O}_i$	Operations that will be executed before $O_i \in \mathcal{O}$ if they are in the same batch.
Binary variable	X_{ik}^b	If operation $O_i \in \mathcal{O}$ is scheduled in the b-th batch of machine $M_k \in \mathcal{M}$.
Binary variable	Y_{gk}^b	If the b-th batch of machine $M_k \in \mathcal{M}$ is of family $F_g \in \mathcal{F}$.
Binary variable	$Z_{i\hat{\imath}}$	If operation O_i and $O_{\hat{\imath}}$ are scheduled in the same batch and O_i precedes $O_{\hat{\imath}}$. (Only for `Batch-S` formulation)
Continuous variable	S_k^b	Starting time of the b-th batch on machine $M_k \in \mathcal{M}$.
Continuous variable	P_k^b	Total processing time of the b-th batch on machine $M_k \in \mathcal{M}$.
Continuous variable	C_i	Completion time of operation $O_i \in \mathcal{O}$.
Continuous variable	C_j	Completion time of job $J_j \in \mathcal{N}$.

B.2
Lower bounds and root node relaxation solution comparison

In this section, we compare the relaxation quality of the developed formulations. In Table B.5, we show the average deviations considering the relaxed solutions of the branch and bound root node (initial lower bounds) and the final lower bounds, after executing the formulations. Information on the number of operations (o) and the number of machines (m) are shown in the first two columns of the table. Deviations are computed as $(Bound - BestBound)/BestBound \times 100$, where $Bound$ is the lower bound generated by each formulation, while *BestBound* is the best lower bound between formulations, in each instance. The results are shown in terms of average deviation and organized by instance group. The last row summarizes the assessment for all 72 instances. Note that, regarding the root node relaxation, `Time-Index` dominates the other formulations, generating the best initial bounds in all groups (average deviation of 0.00%). The bounds generated by `Positional` and `Batch` are 24.03% and 22.61% worse, respectively, considering all instances. Since we are dealing with a minimization problem, negative deviations indicate less tight bounds. In the evaluation of the final lower bounds, `Time-Index` also dominates, providing the best bounds for all instances. In this case, `Positional` is 22.11% worse, while `Batch` is 15.48% worse. It is important to highlight that `Batch` bounds are not valid since the operations are heuristically sequenced within the batches in this formulation, without considering the complete solution space of the problem. However, we are presenting it for comparison. Note that, even limiting the solution space, the relaxation is not good in this formulation, which can be explained by the use of a large number M to compute the completion times of the operations. This condition also affects `Positional` relaxation.

Table B.5: Average deviations between formulations for the root node relaxation solutions and for the final lower bounds on each group of instances.

o	m	Root node relaxation			Final lower bound		
		`Positional`	`Time-Index`	`Batch`†	`Positional`	`Time-Index`	`Batch`†
15	4	-24.20	0.00	-22.50	-17.96	0.00	-10.07
15	8	-21.95	0.00	-19.66	-12.00	0.00	-4.13
25	4	-26.28	0.00	-25.18	-27.24	0.00	-20.05
25	8	-23.88	0.00	-22.08	-25.48	0.00	-14.68
50	4	-23.94	0.00	-23.37	-24.48	0.00	-20.93
50	8	-23.97	0.00	-22.90	-25.50	0.00	-23.03
All		-24.03	0.00	-22.61	-22.11	0.00	-15.48

† `Batch` bounds are not valid since the formulation does not consider the complete solution space of the problem.

B.3 Variables and constraints comparison

In this section, we present a comparison of the number of variables and constraints generated by each of the developed formulations. In Table B.6, we summarize the minimum, average, and the maximum number of variables and constraints for each formulation by instance group. The last row of the table shows the same evaluation for all instances. Note that the number of variables and constraints is lower for `Batch` in all groups, which explains the good performance of the formulation compared to `Time-Index` and `Positional`. `Batch` has 3,506 variables and 7,742 constraints on average, considering all instances. If compared with `Positional`, it means 51.37% fewer variables and 5.56% fewer constraints. Compared to `Time-Index`, the reductions are 95.39% and 71.81%, for variables and constraints, respectively.

Table B.6: Summary of the number of variables and constraints for each of the developed formulations by instance group.

		`Positional`						`Time-Index`						`Batch`					
o	m	Variables			Constraints			Variables			Constraints			Variables			Constraints		
		min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max	min	avg	max
15	4	975	1296	1771	1254	1626	2175	8445	13352	17340	7262	9707	12298	463	618	846	881	1221	1659
	8	1559	2361	2901	2017	2954	3576	19099	25993	34577	14982	19547	24113	729	1113	1372	1255	2149	2665
25	4	2158	2979	3769	2648	3541	4406	17836	29625	37948	10715	14664	17806	1040	1440	1825	2363	3188	3818
	8	4727	5790	6678	5650	6842	7806	45632	59813	71304	24906	29897	34834	2265	2780	3213	4092	5946	7978
50	4	9889	11061	12755	11054	12289	14108	97579	119469	133974	26407	32306	34250	4850	5426	6262	10457	12373	14447
	8	16611	19771	22007	18597	21936	24336	167833	208422	260429	54809	58653	68621	8106	9660	10762	16897	21575	25116
All		975	7210	22007	1254	8198	24336	8445	76112	260429	7262	27462	68621	463	3506	10762	881	7742	25116

B.4 Complete Results for the Mathematical Formulations

In this section, we present, in Table B.7, the total weighted completion time (objective function) found by each formulation for the 72 PLSV instances. The first two columns indicate the name of the instance and the best solution (BEST) found between the formulations (optimal solutions are indicated with an asterisk). For each formulation, the table includes the respective upper bounds (UB), lower bounds (LB), and the gap between them (GAP). We highlighted, in bold, the best solutions found in each instance.

To save space, we shorten the instance names. For example, in the benchmark set, an instance named *PLSV_o15_n5_q3_m4_111_1*, indicates a total of 15 operations (*o15*), associated with 5 jobs (*n5*), divided into 3 families (*q3*), to be scheduled on 4 machines (*m4*). Since the number of jobs depends on the number of operations (calculated as $n = \lfloor o/3 \rfloor$, following the instance generation procedure defined in Chapter 4, we delete it from the instance name. We also suppress the number of families and the last digit because they are equal to 3 and 1 in all instances.

Table B.7: Complete results by instance for each mathematical formulation.

Instance	BEST	Positional			Time-Index			Batch		
		UB	LB	GAP	UB	LB	GAP	UB	LB	GAP
15-4-111	6903*	**6903**	5696.74	17.47	**6903**	6903.00	0.00	6927	6902.00	0.36
15-4-112	7717*	7760	5616.88	27.62	**7717**	7717.00	0.00	7842	6622.00	15.56
15-4-121	7498*	7525	6515.98	13.41	**7498**	7498.00	0.00	7562	7054.00	6.72
15-4-122	8608*	**8608**	6540.74	24.02	**8608**	8608.00	0.00	**8608**	7121.00	17.27
15-4-131	10964*	**10964**	10530.26	3.96	**10964**	10964.00	0.00	**10964**	10914.00	0.46
15-4-132	10685*	**10685**	10667.00	0.17	**10685**	10685.00	0.00	**10685**	10667.00	0.17
15-4-211	8822*	**8822**	6853.69	22.31	**8822**	8822.00	0.00	**8822**	7728.00	12.40
15-4-212	6579*	6801	4224.55	37.88	**6579**	6579.00	0.00	6681	4931.00	26.19
15-4-221	5929*	**5929**	5098.00	14.02	**5929**	5929.00	0.00	**5929**	5236.00	11.69
15-4-222	14545*	14604	11164.11	23.55	**14545**	14545.00	0.00	14726	12240.00	16.88
15-4-231	10755*	10778	9009.04	16.41	**10755**	10755.00	0.00	**10755**	9759.00	9.26
15-4-232	15747*	**15747**	12910.32	18.01	**15747**	15747.00	0.00	**15747**	14404.00	8.53
15-8-111	4306*	**4306**	3939.17	8.52	**4306**	4306.00	0.00	**4306**	4140.00	3.86
15-8-112	5453*	5475	4602.00	15.95	**5453**	5453.00	0.00	**5453**	5082.00	6.80
15-8-121	10790*	**10790**	10790.00	0.00	**10790**	10790.00	0.00	**10790**	10790.00	0.00
15-8-122	8369*	**8369**	7721.00	7.74	**8369**	8369.00	0.00	**8369**	8369.00	0.00
15-8-131	4339*	**4339**	3847.57	11.33	**4339**	4339.00	0.00	**4339**	4231.00	2.49
15-8-132	7371*	7378	6343.72	14.02	**7371**	7371.00	0.00	**7371**	6971.00	5.43
15-8-211	3189*	**3189**	2627.75	17.60	**3189**	3189.00	0.00	3196	3072.00	3.88
15-8-212	4256*	**4256**	3039.00	28.59	**4256**	4256.00	0.00	**4256**	3756.00	11.75
15-8-221	5519*	**5519**	4763.00	13.70	**5519**	5519.00	0.00	**5519**	5063.00	8.26
15-8-222	10461*	**10461**	9720.00	7.08	**10461**	10461.00	0.00	**10461**	9925.00	5.12
15-8-231	8002*	8017	6847.85	14.58	**8002**	8002.00	0.00	**8002**	8002.00	0.00
15-8-232	5127*	5145	4845.00	5.83	**5127**	5127.00	0.00	**5127**	5013.00	2.22
25-4-111	9151	**9151**	5554.34	39.30	9163	8922.14	2.63	9204	6418.00	30.27
25-4-112	18776	19116	8657.18	54.71	19095	16765.52	12.20	**18776**	10097.00	46.22
25-4-121	23134	23398	16521.37	29.39	23508	21676.10	7.79	**23134**	18087.00	21.82
25-4-122	12415	**12415**	9904.01	20.23	**12415**	11979.00	3.51	12423	10788.00	13.16
25-4-131	32800	33396	25556.36	23.47	**32800**	31436.25	4.16	33089	27144.00	17.97
25-4-132	27555*	27642	22707.04	17.85	**27555**	27555.00	0.00	27834	24242.00	12.91
25-4-211	30098	30259	15388.35	49.14	**30098**	26987.68	10.33	30168	17471.84	42.08
25-4-212	20012	**20012**	9720.21	51.43	20172	17661.94	12.44	20412	10982.00	46.20

Continued on next page

Table B.7 – continued from previous page

Instance	BEST	Positional			Time-Index			Batch		
		UB	LB	GAP	UB	LB	GAP	UB	LB	GAP
25-4-221	19944	20461	14420.32	29.52	20573	19004.43	7.62	**19944**	15431.00	22.63
25-4-222	27274	**27274**	18924.14	30.61	27666	24502.00	11.44	27306	22495.00	17.62
25-4-231	29552	29579	24569.09	16.94	**29552**	28908.83	2.18	29727	25365.00	14.67
25-4-232	29502	29544	24095.50	18.44	**29502**	27874.68	5.52	29605	25579.00	13.60
25-8-111	11558	11754	7020.29	40.27	11561	10985.29	4.98	**11558**	8879.00	23.18
25-8-112	16150	16153	8677.00	46.28	**16150**	15041.61	6.86	16397	13286.00	18.97
25-8-121	10478	10678	8037.00	24.73	10549	10416.18	1.26	**10478**	9116.00	13.00
25-8-122	19723	20154	14262.70	29.23	**19723**	18353.77	6.94	19900	14878.00	25.24
25-8-131	9700*	9992	7521.45	24.73	**9700**	9699.82	0.00	9753	8556.00	12.27
25-8-132	17947	18385	14525.00	21.00	**17947**	17613.80	1.86	18069	16304.00	9.77
25-8-211	8369	8399	5707.00	32.05	**8369**	8038.05	3.95	8454	6947.00	17.83
25-8-212	13518	13718	8439.00	38.48	**13518**	12741.30	5.75	14074	8904.00	36.73
25-8-221	10993	**10993**	8602.07	21.75	**10993**	10973.35	0.18	11004	9879.00	10.22
25-8-222	14140	14759	9818.00	33.48	**14140**	13003.73	8.04	14397	10140.00	29.57
25-8-231	9251	9260	7467.00	19.36	**9251**	9112.00	1.50	9401	8551.00	9.04
25-8-232	16929	17104	13922.00	18.60	**16929**	16430.36	2.95	17179	14319.00	16.65
50-4-111	64068	70430	31231.83	55.66	77915	53498.00	31.34	**64068**	35047.00	45.30
50-4-112	102246	105420	49573.18	52.98	141654	81876.00	42.20	**102246**	48787.00	52.28
50-4-121	65231	66821	49800.84	25.47	79298	59448.00	25.03	**65231**	52761.00	19.12
50-4-122	109078	111945	74609.00	33.35	139816	93961.00	32.80	**109078**	78618.00	27.93
50-4-131	92825	95613	77325.33	19.13	103406	86551.00	16.30	**92825**	79871.00	13.96
50-4-132	99080	101816	83992.55	17.51	108063	91768.00	15.08	**99080**	86368.00	12.83
50-4-211	84383	**84383**	36992.27	56.16	101148	72437.00	28.39	84445	40150.00	52.45
50-4-212	87761	89998	40072.31	55.47	116313	69962.00	39.85	**87761**	42139.00	51.98
50-4-221	106430	107875	73157.28	32.18	128244	94499.00	26.31	**106430**	77429.00	27.25
50-4-222	105136	108737	64687.01	40.51	130840	84786.00	35.20	**105136**	68495.00	34.85
50-4-231	89865	90592	77028.57	14.97	92482	82375.00	10.93	**89865**	80115.00	10.85
50-4-232	131034	133575	99852.99	25.25	162223	113917.00	29.78	**131034**	101660.00	22.42
50-8-111	34381	35214	19914.72	43.45	42281	30308.00	28.32	**34381**	22974.00	33.18
50-8-112	50521	**50521**	23712.00	53.07	63573	37204.00	41.48	50809	24163.00	52.44
50-8-121	31944	32317	22607.25	30.05	35628	28021.00	21.35	**31944**	23330.00	26.97
50-8-122	42961	44135	27248.00	38.26	55892	35185.00	37.05	**42961**	27689.00	35.55
50-8-131	60738	61587	48452.00	21.33	63390	56573.00	10.75	**60738**	49247.00	18.92

Continued on next page

Table B.7 – continued from previous page

Instance	BEST	Positional			Time-Index			Batch		
		UB	LB	GAP	UB	LB	GAP	UB	LB	GAP
50-8-132	65103	66941	46238.41	30.93	73496	53436.00	27.29	**65103**	49403.00	24.12
50-8-211	48355	49758	24093.00	51.58	59431	41047.00	30.93	**48355**	24109.00	50.14
50-8-212	60291	61043	26170.00	57.13	75680	45357.00	40.07	**60291**	26635.00	55.82
50-8-221	35471	36556	24375.00	33.32	41663	31017.00	25.55	**35471**	24741.00	30.25
50-8-222	56600	57164	32053.95	43.93	68310	45464.00	33.44	**56600**	32603.00	42.40
50-8-231	54080	55899	45961.00	17.78	59774	50996.00	14.69	**54080**	46400.00	14.20
50-8-232	62858	64263	45023.07	29.94	67058	57253.00	14.62	**62858**	46664.00	25.76

C
Constructive Heuristics

C.1
Results by Instance for the constructive heuristics

In this section, we present the total weighted completion time (objective function) found by each constructive heuristic introduced in Chapter 5 for the benchmark of 72 PLSVSP instances (Table C.1). The first two columns indicate the name of the instance and the best solution found among the heuristics. To save space, we shorten the instance names as described in Appendix B.4.

Table C.1: Complete results for the constructive heuristics.

Instance	Best	ERD	SPT	LPT	MCT	WSPT -MAX-	WSPT -SUM-	WSPT -AVG-	WSPT -WAVG-	WSPT -WAVGA-	WMCT -MAX-	WMCT -SUM-	WMCT -AVG-	WMCT -WAVG-	WMCT -WAVGA-	WMCT-Pair -MAX-	WMCT-Pair -SUM-	WMCT-Pair -AVG-	WMCT-Pair -WAVG-	WMCT-Pair -WAVGA-
15-4-111	7444	9095	10713	11381	9364	7462	7462	8274	7997	8897	7618	7444	7618	7444	8007	7618	7444	7618	7444	8047
15-4-112	8551	11495	10318	11837	11348	9473	10236	10243	10596	10181	9527	9199	9469	9686	8551	9527	9199	9469	9686	8551
15-4-121	7969	9665	10938	13180	10499	10884	10884	11244	9019	7969	9524	9093	9425	8512	8512	9715	9093	9425	8431	8782
15-4-122	9799	9799	12461	13682	11765	12067	12018	11679	10876	11417	11190	11038	11092	9848	10175	11190	11038	11092	9888	10175
15-4-131	11094	13864	15040	14549	15040	13060	13308	13060	12557	11472	12040	11831	12328	11094	11094	12040	11831	12328	11094	11154
15-4-132	11948	12919	14861	13936	13044	13595	13668	14099	13744	13217	11948	12198	12343	12373	12308	11948	12198	12343	12462	12462
15-4-211	9677	11668	11742	13026	11662	11282	11123	11282	10160	9767	9961	10441	10446	11321	9677	9961	10441	10446	11321	9677
15-4-212	7037	8917	10525	10601	9805	9307	8383	8993	8106	7617	8416	7538	7969	8397	7037	8500	7538	7969	8236	7037
15-4-221	6608	8405	8012	8399	7983	7476	7250	7559	7524	6965	6619	6633	7101	6813	6813	6619	6633	7101	6608	6813
15-4-222	15948	22350	19317	18987	19255	18086	17544	17265	18170	16728	17724	17911	18497	16695	15948	17724	17911	18497	16695	15948

Continued on next page

Table C.1 – continued from previous page

Instance	Best	ERD	SPT	LPT	MCT	WSPT -MAX-	WSPT -SUM-	WSPT -AVG-	WSPT -WAVG-	WSPT -WAVGA-	WMCT -MAX-	WMCT -SUM-	WMCT -AVG-	WMCT -WAVG-	WMCT -WAVGA-	WMCT-Pair -MAX-	WMCT-Pair -SUM-	WMCT-Pair -AVG-	WMCT-Pair -WAVG-	WMCT-Pair -WAVGA-
15-4-231	12002	13727	13534	14614	13034	13037	13444	13438	13233	12502	13179	12486	13884	12545	12002	13179	12486	13884	12545	12104
15-4-232	17847	20374	21169	21341	18869	19006	19326	19846	19006	19462	18027	18047	18184	18756	18436	17847	18005	18120	18696	18436
15-8-111	4586	5623	4964	5635	5095	5118	4855	5283	4760	4629	4690	4795	5169	4586	4795	4687	4795	5163	4586	4795
15-8-112	6064	7371	7242	6882	7251	6392	6392	6217	6392	6271	6147	6147	6151	6155	6064	6112	6112	6067	6143	6100
15-8-121	11195	11195	18692	11290	13544	13904	12435	13904	11751	11863	11195	11667	11195	11894	11195	11195	11667	11195	11894	11195
15-8-122	8929	11366	8929	9146	9113	9074	10456	8929	10864	9938	9749	10778	9016	9685	9209	9749	10778	9016	9685	9209
15-8-131	4603	6178	5812	5656	5509	4840	4840	5656	5053	4603	4798	4798	4702	4999	4633	4798	4798	4702	4999	4633
15-8-132	7723	8508	8173	9631	8108	7723	7863	7731	7787	7913	7787	7787	7754	7731	7731	7787	7787	7754	7731	7731
15-8-211	3396	4650	4233	3895	4176	3552	3552	3552	3581	3468	3582	3589	3582	3418	3396	3582	3589	3582	3406	3430
15-8-212	4342	5605	5626	5815	5619	4671	4715	4581	4705	4665	4749	4646	4342	4732	4646	4749	4568	4342	4715	4568
15-8-221	6475	7352	8000	7630	7158	6727	6727	6845	7055	6891	6801	6801	6475	6820	6860	6801	6801	6563	6820	6860
15-8-222	12337	13781	13585	14499	13585	15990	13479	13819	12873	12950	12750	12337	13343	12665	12511	13124	12337	13343	12665	12665
15-8-231	8110	9036	9185	8206	9261	8134	8134	8134	9086	8372	8134	8134	8134	8110	8309	8134	8134	8134	8110	8309
15-8-232	5454	6483	7218	6072	7233	5469	5778	5970	5778	5505	5832	5505	5466	5505	5454	5898	5454	5478	5454	5529
25-4-111	10263	15292	17505	17568	15435	12326	12057	13125	12344	11555	11173	11582	11997	10295	10263	11173	11582	11997	10295	10471
25-4-112	21655	27075	27488	33343	30831	25167	25723	25857	24620	23075	22554	24125	22323	22668	21842	21655	24125	22323	22668	21926
25-4-121	25371	35202	33992	36125	31063	29933	29172	29996	29045	27071	26121	27588	26871	25668	26632	25371	27588	26015	25748	26632
25-4-122	13905	18604	20013	19208	18008	17114	16874	18902	17629	17385	14121	15104	17895	13905	14163	14121	14782	16606	13905	14155
25-4-131	37019	42554	44623	49718	40155	43673	44628	41967	41089	39296	37019	38799	37481	38254	37828	37019	38656	37460	38254	38058
25-4-132	31571	33911	40801	45725	34202	38003	39559	41636	39928	35033	32217	34674	34873	32636	31571	32217	34674	35153	32072	32281
25-4-211	32367	44982	44580	51040	43400	42699	35517	40056	36627	32987	37544	35848	37662	35890	32367	37427	35662	38449	37393	34148
25-4-212	23303	29192	29934	31132	29002	27214	25979	29533	27875	25658	23353	27193	26524	25819	24641	23303	26263	28086	26131	24367
25-4-221	22267	29642	33937	32374	30899	29540	27733	28970	27055	26328	27669	25653	26930	23652	22267	27484	25653	26930	23839	22581
25-4-222	29749	31390	37468	33553	35605	33430	32620	38308	33240	33617	31953	30819	32481	30912	29749	31953	30819	32481	30912	29893
25-4-231	32329	38344	43323	42292	42184	41837	41337	41147	40656	37826	34740	35001	35950	33068	32329	34740	34906	35950	33068	32862
25-4-232	35049	36134	46387	45360	39399	45616	40586	47534	40885	37147	38086	35366	38644	35805	35049	38183	35366	39278	35805	35049
25-8-111	12438	14305	16780	16879	16521	15645	14709	15825	14906	13376	14846	14155	15450	12816	12884	14846	14155	15355	12816	12438
25-8-112	19767	21947	21283	23152	21231	22201	21225	22141	21577	21926	19767	20463	21469	22255	20044	19962	20463	21469	22255	20044
25-8-121	11367	14409	17130	15067	16169	14428	13414	14322	11867	11635	11643	11745	11866	11409	11367	11753	11745	11800	11494	11367
25-8-122	21984	24377	25257	25050	24363	23428	23062	24814	24425	22029	22776	24236	24218	24000	21984	22776	24236	23794	24000	22002
25-8-131	10001	14225	13626	15973	13527	12074	11657	12374	11295	10932	10572	10001	11967	10431	10505	10572	10001	11382	10502	10558
25-8-132	19133	21787	24434	22067	23196	24173	22211	22719	21502	20171	22511	21118	22226	19656	19133	21719	21118	21616	19656	19363
25-8-211	9393	12982	13825	13462	14637	10682	10339	10571	10023	9806	9899	10035	9680	9847	9393	10023	9723	9778	9687	9395
25-8-212	15149	16537	18523	19791	18228	18779	18656	19335	17190	16173	18621	19517	19184	16544	15149	18621	18163	19009	15911	15562
25-8-221	12185	14919	17180	16228	15538	14650	14521	14714	14836	14484	13910	13245	13604	13613	12185	13910	13245	13523	13035	12338

Continued on next page

Table C.1 – continued from previous page

Instance	Best	ERD	SPT	LPT	MCT	WSPT -MAX-	WSPT -SUM-	WSPT -AVG-	WSPT -WAVG-	WSPT -WAVGA-	WMCT -MAX-	WMCT -SUM-	WMCT -AVG-	WMCT -WAVG-	WMCT -WAVGA-	WMCT-Pair -MAX-	WMCT-Pair -SUM-	WMCT-Pair -AVG-	WMCT-Pair -WAVG-	WMCT-Pair -WAVGA-
25-8-222	15772	17208	20235	17468	17947	18166	18245	18302	17767	16318	17669	16915	17362	16198	15858	16811	16791	17362	16198	15772
25-8-231	10157	11481	12339	10970	11378	10645	10928	11868	11136	10157	10183	10473	10823	10263	10180	10183	10473	10823	10165	10272
25-8-232	17813	20610	21909	20968	21123	19677	20062	19493	20489	19789	18118	18279	17813	19401	19230	18118	18279	17929	19401	19230
50-4-111	74774	108184	111179	129960	99840	91771	98297	98466	93228	82488	77235	88247	94768	86220	78466	74774	86880	88894	84173	79278
50-4-112	113765	145788	154836	153691	151717	144916	137004	144686	130922	119877	118963	127386	132296	125817	116919	116804	128155	128254	122312	113765
50-4-121	75727	101557	114354	119339	94268	102667	101916	103746	94912	87680	84837	88155	88798	80250	75727	84328	88110	88409	79643	75843
50-4-122	123813	154439	172323	191179	152129	158905	157955	162322	153978	143462	133152	149718	147096	140083	123813	128786	150146	145600	140083	123881
50-4-131	105604	130939	168283	145498	130204	143675	142495	143062	124464	121436	105604	119512	109447	116412	116498	106984	119248	108647	116412	117427
50-4-132	107780	133506	165648	154187	135527	161387	160165	157576	155081	145316	110442	132379	122785	123647	114357	107780	132404	122785	124296	113429
50-4-211	88068	121234	131463	135979	125447	113382	107610	121532	109450	98578	98004	96903	103978	96857	88068	98169	95908	102390	95728	88068
50-4-212	91988	130181	139964	144482	125511	129884	116569	127136	114736	105529	101005	109391	112401	99147	92556	96773	103666	111811	99196	91988
50-4-221	129455	162268	171369	184071	148045	149295	152045	158909	144899	133614	129771	157278	134604	137834	129682	131853	158772	134604	137834	129455
50-4-222	122095	133334	175586	154759	148554	156027	156152	164236	155422	142770	122095	142812	139128	143825	128304	123513	141656	139443	143287	128711
50-4-231	105334	112385	143965	166122	114861	127941	130149	131960	129579	118576	105334	110289	115546	115395	108208	106467	110588	115442	109627	108744
50-4-232	144658	163796	185888	181001	169699	179771	175133	181582	171171	157407	149780	165183	167808	161031	144661	148384	165183	165414	161031	144658
50-8-111	36764	55699	54547	64875	50867	44284	45075	45096	43275	42647	38490	40783	40225	39913	36764	38653	39905	40036	39239	37260
50-8-112	50915	68412	67045	70152	61249	65968	61328	66463	57451	54461	61761	59657	64491	55958	51512	61338	59288	63278	55537	50915
50-8-121	38062	44553	56482	54158	51716	44897	44443	46020	43537	40729	40143	40343	40232	38717	38062	40123	40336	40221	38717	38131
50-8-122	46364	55933	69336	58323	61350	58852	58907	62062	57208	53222	48859	51033	51687	49314	46364	49213	50293	51453	49315	46364
50-8-131	68008	77405	92697	91720	80622	86045	83778	89643	81077	73340	69379	75060	76571	73940	68008	70207	75290	76853	73940	69421
50-8-132	68833	78371	100479	82661	88875	94329	89336	94963	83000	79633	78562	76500	77251	75477	69662	79178	78097	77520	73063	68833
50-8-211	50009	66338	68135	76791	68205	64621	62416	64455	60205	54628	59005	59838	60807	56043	50520	58022	59173	55779	55611	50009
50-8-212	60848	77535	77491	81317	78535	72394	74184	73220	72552	63566	67058	71200	71149	70600	60848	65670	69974	71655	70516	60974
50-8-221	37090	52511	51315	53566	49627	43219	43746	48524	42549	38954	40077	42505	43371	38226	37090	39988	42021	43387	38160	37373
50-8-222	61897	70389	76352	74470	71178	73447	75879	74374	70175	67386	67125	67618	66752	68295	62205	64778	66386	65320	66949	61897
50-8-231	60485	68314	86694	87497	80227	78004	76486	82500	75266	64781	64779	66756	69242	62092	60485	65047	66662	69197	62272	60492
50-8-232	69575	83868	98096	100155	92059	94037	93982	96221	93828	93181	77229	84392	84036	77676	69575	76513	84256	84159	77592	70452

D
Matheuristics

D.1
Matheuristics Comparative Analysis

In this section, we compare the results for the matheuristic variations. In Table D.1, the results for the `ILS-Math` variations are shown in terms of the average RPD ($\overline{RPD}$) and average computational time ($\overline{time}$). The same results but for the `GRASP-Math` variations are presented in Table D.2.

Table D.1: Average results for `ILS-Math` variations.

o	m	$\texttt{ILS-Math}_1$		$\texttt{ILS-Math}_2$		$\texttt{ILS-Math}_3$	
		$\overline{RPD}$	$\overline{time}$	$\overline{RPD}$	$\overline{time}$	$\overline{RPD}$	$\overline{time}$
15	4	0.61	**14**	**0.33**	18	0.42	**14**
	8	0.11	**46**	**0.03**	49	0.08	**46**
25	4	0.06	**123**	-0.13	143	**-0.17**	**123**
	8	-0.15	247	-0.18	267	**-0.20**	**244**
50	4	-2.78	1054	-2.74	**984**	**-3.02**	1052
	8	-3.99	**972**	-3.86	994	**-4.09**	990
All instances		-1.02	**409**	-1.09	**409**	**-1.16**	412

Table D.2: Average results for `GRASP-Math` variations.

o	m	$\texttt{GRASP-Math}_1$		$\texttt{GRASP-Math}_2$		$\texttt{GRASP-Math}_3$	
		$\overline{RPD}$	$\overline{CPU}$	$\overline{RPD}$	$\overline{CPU}$	$\overline{RPD}$	$\overline{CPU}$
15	4	0.49	**25**	**0.29**	28	0.30	26
15	8	0.05	**53**	0.02	65	**0.01**	54
25	4	-0.15	**184**	**-0.38**	217	-0.31	186
25	8	-0.20	298	**-0.27**	313	**-0.27**	**297**
50	4	-2.34	**1140**	-2.50	1194	**-2.65**	1159
50	8	-3.74	**1038**	**-3.86**	1173	-3.80	1066
All instances		-0.98	**456**	-1.117	498	**-1.119**	465

Note that results for the `ILS-Math` are quite similar for all variations. However, we can see an advantage for the $\texttt{ILS-Math}_3$ approach when the size of the instances increases (instances with 25 and 50 operations). Regarding the complete set of instances, $\texttt{ILS-Math}_3$ advantage is clearer with it dominating the other approaches in terms of the $\overline{RPD}$ with -1.16%. Similar behavior can be noted for the `GRASP-Math` variations. $\texttt{GRASP-Math}_3$ presented the best overall $\overline{RPD}$ of -1.119% with a small difference in the average computational time for the best approach in this criterion (465 seconds against 456 seconds for the $\texttt{GRASP-Math}_1$).

D.2
Results by Instance for the Matheuristics

In this section, we present the total weighted completion time (objective function) found by each method introduced in Chapter 6 (`Batch-WSPT`, `Batch-S`, `ILS-Math`$_3$, and `GRASP-Math`$_3$) for the benchmark of 72 PLSVSP instances with the respective computational times (Table D.3). The first two columns indicate the name of the instance and its BKS found while running the pure mathematical formulations (optimal solutions are indicated with an asterisk), described in Chapter 4. Then, the upper bound (ub) and the computational time (time) for each formulation are presented, followed by the results of each matheuristic variation. The matheuristic results are presented in the following order: minimum total weighted completion time (min), average total weighted completion time (avg), and average computational time ($\overline{time}$), among the runs. We highlighted, in bold, the best solution for each instance. We suppress computational times that have reached the time limit (21600 seconds) defined for the MIP formulations. To save space, we shorten the instance names as described in Appendix B.4.

Table D.3: Complete results by instance regarding the PLSVSP benchmark.

Instance	BKS	Batch-WSPT		Batch-S		ILS-Math$_1$			ILS-Math$_2$			ILS-Math$_3$			GRASP-Math$_1$			GRASP-Math$_2$			GRASP-Math$_3$		
		ub	time	ub	time	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$
15-4-111	6903*	6927	12417	**6903**	12121	6927	6930.8	2	**6903**	6903.0	4	**6903**	6909.2	2	**6903**	6915.0	5	**6903**	6903.0	4	**6903**	6903.0	5
15-4-112	7717*	7842	-	**7717**	-	7760	7763.6	21	**7717**	7755.3	26	7760	7763.6	21	**7717**	7742.6	42	**7717**	7736.2	46	**7717**	7742.6	43
15-4-121	7498*	7589	-	**7498**	-	7589	7609.6	5	**7498**	7539.6	8	**7498**	7532.4	5	7589	7595.1	7	**7498**	7500.7	10	**7498**	7498.0	8
15-4-122	8608*	**8608**	-	**8608**	-	**8608**	8608.0	10	**8608**	8612.0	15	**8608**	8608.0	10	**8608**	8608.0	18	**8608**	8608.0	26	**8608**	8608.0	18
15-4-131	10964*	**10964**	16546	**10964**	20576	**10964**	10996.0	2	**10964**	10983.0	2	**10964**	10996.0	2	**10964**	11222.5	0	**10964**	11292.5	0	**10964**	11222.5	0
15-4-132	10685*	**10685**	-	**10685**	-	**10685**	10685.0	5	**10685**	10685.0	6	**10685**	10685.0	5	**10685**	10687.1	10	**10685**	10687.1	11	**10685**	10687.1	10
15-4-211	8822*	**8822**	-	**8822**	-	**8822**	8872.0	12	**8822**	8859.5	11	**8822**	8872.0	12	**8822**	8822.0	24	**8822**	8834.5	19	**8822**	8822.0	24
15-4-212	6579*	6681	-	**6579**	-	6681	6684.6	49	**6579**	6680.8	61	**6579**	6628.2	51	**6579**	6611.0	104	**6579**	6579.0	89	**6579**	6579.0	109
15-4-221	5929*	**5929**	-	**5929**	-	**5929**	5929.0	11	**5929**	5929.0	14	**5929**	5929.0	11	**5929**	5929.0	18	**5929**	5929.0	25	**5929**	5929.0	19
15-4-222	14545*	14712	-	**14545**	-	14712	14754.6	18	**14545**	14621.9	21	14712	14754.6	18	**14545**	14628.8	27	**14545**	14545.0	40	**14545**	14628.8	27
15-4-231	10755*	**10755**	-	**10755**	-	**10755**	10859.3	15	**10755**	10781.5	25	**10755**	10845.7	16	**10755**	10817.6	26	**10755**	10764.1	36	**10755**	10790.4	26
15-4-232	15747*	**15747**	-	**15747**	-	**15747**	15747.0	13	**15747**	15747.0	17	**15747**	15747.0	13	**15747**	15758.2	22	**15747**	15747.0	28	**15747**	15747.0	23

Continued on next page

Table D.3 – continued from previous page

Instance	BKS	Batch-WSPT		Batch-S		ILS-Math$_1$			ILS-Math$_2$			ILS-Math$_3$			GRASP-Math$_1$			GRASP-Math$_2$			GRASP-Math$_3$		
		ub	time	ub	time	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$
15-8-111	4306*	**4306**	-	**4306**	-	**4306**	4306.0	32	**4306**	4306.0	42	**4306**	4306.0	32	**4306**	4306.0	44	**4306**	4306.0	56	**4306**	4306.0	45
15-8-112	5453*	**5453**	-	**5453**	-	**5453**	5453.0	73	**5453**	5453.0	97	**5453**	5453.0	73	**5453**	5453.0	92	**5453**	5453.0	117	**5453**	5453.0	93
15-8-121	10790*	**10790**	0	**10790**	1	**10790**	10790.0	1	**10790**	10790.0	2	**10790**	10790.0	1	**10790**	10790.0	1	**10790**	10790.0	2	**10790**	10790.0	1
15-8-122	8369*	**8369**	2	**8369**	2	**8369**	8369.0	9	**8369**	8369.0	10	**8369**	8369.0	9	**8369**	8369.0	19	**8369**	8369.0	17	**8369**	8369.0	19
15-8-131	4339*	**4339**	-	**4339**	-	**4339**	4339.0	10	**4339**	4339.0	13	**4339**	4339.0	10	**4339**	4339.0	14	**4339**	4339.0	15	**4339**	4339.0	14
15-8-132	7371*	**7371**	-	**7371**	-	**7371**	7386.3	125	**7371**	7379.8	123	**7371**	7386.3	125	**7371**	7382.3	132	**7371**	7383.1	136	**7371**	7382.9	132
15-8-211	3189*	3203	-	**3189**	-	3203	3203.3	72	**3189**	3191.8	69	**3189**	3193.2	74	3203	3203.0	74	**3189**	3189.0	109	**3189**	3189.0	78
15-8-212	4256*	**4256**	-	4263	-	**4256**	4256.7	112	**4256**	4258.5	97	**4256**	4256.7	112	**4256**	4256.0	111	**4256**	4257.4	133	**4256**	4256.0	111
15-8-221	5519*	**5519**	-	**5519**	-	**5519**	5519.0	26	**5519**	5522.0	30	**5519**	5519.0	26	**5519**	5519.0	35	**5519**	5519.0	44	**5519**	5519.0	36
15-8-222	10461*	**10461**	-	**10461**	-	**10461**	10530.3	7	**10461**	10461.0	8	**10461**	10530.3	7	**10461**	10461.0	9	**10461**	10461.0	11	**10461**	10461.0	9
15-8-231	8002*	**8002**	-	**8002**	-	**8002**	8002.0	36	**8002**	8005.0	35	**8002**	8002.0	36	**8002**	8002.0	55	**8002**	8003.5	68	**8002**	8002.0	55
15-8-232	5127*	**5127**	-	**5127**	-	**5127**	5127.0	50	**5127**	5127.0	60	**5127**	5127.0	51	**5127**	5127.0	54	**5127**	5127.0	69	**5127**	5127.0	54
25-4-111	9151	9192	-	**9151**	-	9192	9217.3	118	9166	9237.1	102	**9151**	9176.1	119	**9151**	9174.3	150	**9151**	9157.3	161	**9151**	9152.2	150
25-4-112	18776	18896	-	19012	-	18752	18795.2	151	18734	18817.7	174	**18678**	18751.8	153	18752	18790.9	192	**18678**	18707.4	222	**18678**	18743.5	195
25-4-121	23134	23364	-	23365	-	22927	23080.1	155	**22865**	23011.0	253	**22865**	23050.7	161	**22865**	23039.0	263	**22865**	23026.9	331	**22865**	22986.4	294
25-4-122	12415	12417	-	**12415**	-	12417	12417.6	51	**12415**	12415.6	52	**12415**	12416.8	51	**12415**	12429.9	77	**12415**	12415.0	90	**12415**	12422.8	78
25-4-131	32800	33092	-	32966	-	32854	32862.3	176	**32800**	32859.9	215	32854	32862.3	176	**32800**	32848.7	314	32820	32863.4	224	**32800**	32860.9	295
25-4-132	27555*	28036	-	**27555**	-	27651	27663.2	93	**27555**	27608.2	109	**27555**	27555.0	88	**27555**	27633.0	188	**27555**	27562.4	234	**27555**	27555.0	187
25-4-211	30098	30534	-	30301	-	30002	30103.6	132	**29679**	29924.1	184	30002	30099.3	133	29759	29794.8	180	**29679**	29710.2	220	**29679**	29766.2	181
25-4-212	20012	19669	-	20667	-	19669	19700.3	129	**19595**	19639.4	146	**19595**	19638.8	131	19628	19719.2	188	**19595**	19607.4	310	**19595**	19694.9	191
25-4-221	19944	20399	-	20637	-	19944	19970.3	120	**19833**	19955.7	148	**19833**	19929.8	124	19944	20018.9	155	**19833**	19945.3	225	**19833**	19987.3	153
25-4-222	27274	27220	-	27268	-	27307	27360.2	60	**27148**	27267.7	107	27185	27262.1	61	27155	27219.0	80	**27148**	27165.9	162	**27148**	27184.7	77
25-4-231	29552	29695	-	**29552**	-	29695	29698.2	91	**29552**	29580.6	81	**29552**	29552.0	89	**29552**	29638.1	174	**29552**	29552.0	155	**29552**	29552.0	186
25-4-232	29502	29894	-	**29383**	-	29540	29552.0	201	**29383**	29431.2	144	**29383**	29472.7	186	**29383**	29407.1	251	**29383**	29408.8	273	**29383**	29400.8	245
25-8-111	11558	11634	-	11479	-	**11387**	11464.1	204	**11387**	11447.5	255	**11387**	11464.1	205	11427	11465.1	361	**11387**	11444.4	334	11427	11453.0	363
25-8-112	16150	16191	-	16295	-	16165	16267.8	309	16056	16203.4	273	16061	16180.9	300	**16053**	16123.8	373	**16053**	16095.7	389	**16053**	16089.4	354
25-8-121	10478	**10478**	-	10571	-	**10478**	10537.1	282	**10478**	10539.0	278	**10478**	10548.8	254	**10478**	10544.9	287	**10478**	10541.9	294	**10478**	10548.5	309
25-8-122	19723	19802	-	19778	-	**19647**	19658.0	154	**19647**	19659.4	187	**19647**	19658.0	150	**19647**	19661.5	177	**19647**	19668.6	206	**19647**	19659.5	177
25-8-131	9700*	9716	-	**9700**	-	**9700**	9700.0	145	**9700**	9702.2	151	**9700**	9700.0	145	**9700**	9700.9	183	**9700**	9700.6	184	**9700**	9700.0	185
25-8-132	17947	**17911**	-	17989	-	**17911**	17931.0	198	**17911**	17941.6	223	**17911**	17934.8	194	**17911**	17964.6	214	**17911**	17946.4	265	**17911**	17956.2	225
25-8-211	8369	8367	-	8429	-	**8261**	8315.2	307	**8261**	8316.1	383	**8261**	8325.6	384	8308	8334.3	405	**8261**	8327.1	391	8294	8328.9	372
25-8-212	13518	13675	-	13963	-	13397	13539.1	298	13388	13505.4	378	13397	13530.6	275	**13337**	13472.4	432	**13337**	13454.3	417	**13337**	13454.1	413
25-8-221	10993	11026	-	11004	-	11019	11029.4	150	**10993**	11036.0	151	**10993**	11012.9	150	11004	11025.0	144	**10993**	11000.8	189	**10993**	11006.7	148

Continued on next page

Table D.3 – continued from previous page

Instance	BKS	Batch-WSPT		Batch-S		ILS-Math$_1$			ILS-Math$_2$			ILS-Math$_3$			GRASP-Math$_1$			GRASP-Math$_2$			GRASP-Math$_3$		
		ub	time	ub	time	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$	min	avg	$\overline{time}$
25-8-222	14140	13998	-	14001	-	**13819**	13937.4	345	**13819**	13973.6	360	13845	13934.4	327	13902	13968.9	314	13845	13947.0	422	13844	13939.0	365
25-8-231	9251	9312	-	9282	-	**9230**	9232.5	241	**9230**	9237.6	229	**9230**	9231.4	239	**9230**	9233.9	348	**9230**	9241.8	265	**9230**	9238.2	338
25-8-232	16929	17005	-	17324	-	16867	16913.5	325	**16863**	16916.8	337	**16863**	16906.2	304	16889	16930.2	333	16880	16937.9	395	16879	16932.3	316
50-4-111	64068	68673	-	70886	-	61288	61935.1	1381	61237	61943.4	1238	61179	61697.1	1159	61346	62766.2	1311	61577	62328.9	1541	**61023**	62129.3	1242
50-4-112	102246	102201	-	102283	-	98283	98719.3	1165	98116	98668.3	1127	**97260**	98259.4	1010	98093	98912.2	1511	98332	99184.0	1082	98582	99068.3	1245
50-4-121	65231	65185	-	65367	-	63447	63728.6	1075	63578	63947.6	820	**63395**	63648.4	1038	63736	64064.0	1170	63693	64103.4	1018	63752	64133.9	974
50-4-122	109078	110531	-	108577	-	106608	107269.2	1067	106194	107162.2	895	**106009**	106811.5	1418	106776	107705.8	1153	106388	107780.3	1044	106491	107224.0	1229
50-4-131	92825	92011	-	92386	-	91497	91748.6	1166	**91341**	91698.6	1075	91481	91751.8	831	91532	92005.7	1047	91566	91803.9	1034	91398	91675.4	1054
50-4-132	99080	99690	-	99599	-	**97906**	98372.1	818	98343	98593.0	803	**97906**	98448.1	601	98208	98858.8	852	98382	98782.8	1095	98208	98760.3	1109
50-4-211	84383	82439	-	81604	-	80032	80469.9	1061	80254	80686.1	1160	79436	80281.2	1110	79792	80779.7	1193	79611	80506.8	1390	**78988**	80475.2	1333
50-4-212	87761	83410	-	86607	-	81350	81920.5	805	81448	82028.5	1110	81145	81713.8	934	81881	82204.4	1210	81678	81995.5	1453	**80798**	81669.1	1308
50-4-221	106430	105371	-	104586	-	103930	104350.2	1011	103516	104005.3	1086	**103345**	104019.3	1125	103732	104368.3	1449	103496	104426.2	1373	103538	104292.5	1232
50-4-222	105136	104714	20784†	107833	-	101723	102670.7	1177	101794	102557.0	768	**100921**	102190.6	1215	102113	102938.4	959	101563	102465.9	914	101225	102343.6	1234
50-4-231	89865	85913	-	87968	-	85588	85814.1	890	85401	85746.2	912	**85290**	85701.7	1062	85456	85827.3	797	85475	85839.8	1401	85393	85923.0	892
50-4-232	131034	133193	2948†	134424	-	129575	130260.5	1032	129334	130449.9	808	**128913**	129813.0	1125	130160	131671.3	1028	129823	131231.3	979	129649	130932.7	1056
50-8-111	34381	33354	-	35691	-	**33248**	33601.0	989	33402	33669.0	1029	33269	33514.7	997	33672	33804.2	933	33590	33751.7	901	33574	33800.8	882
50-8-112	50521	47910	-	48461	-	45320	45815.3	1074	45596	46078.9	899	**44980**	45704.9	1177	45906	46149.1	1191	45965	46225.8	1521	45609	46067.1	1676
50-8-121	31944	30906	-	32771	6026†	30393	30564.7	786	**30276**	30493.0	977	30349	30593.1	891	30332	30618.4	832	30321	30541.1	849	30336	30591.2	870
50-8-122	42961	41870	-	43002	-	41111	41419.7	890	41101	41496.5	875	41147	41379.2	1196	41178	41389.9	1077	**41080**	41339.6	1095	41204	41360.6	1173
50-8-131	60738	59588	-	60242	-	**59018**	59450.0	673	59377	59538.9	857	59111	59483.5	765	59276	59684.5	968	59410	59651.2	951	59335	59629.8	1096
50-8-132	65103	64953	17470†	68600	4164†	62657	63114.0	997	62797	63071.0	942	62486	62828.3	805	**62387**	63022.3	1118	62665	63116.7	988	62758	63019.4	1118
50-8-211	48355	48884	-	50349	-	**46323**	46927.1	1207	46664	46987.4	1258	46489	46861.7	1102	46589	47007.7	1400	46586	46971.6	1390	46510	47026.8	1178
50-8-212	60291	58338	6810†	60588	5296†	55215	55913.3	1258	55512	56081.6	1268	55375	55858.9	1417	**54994**	55895.1	1393	55372	55711.6	1586	55528	55873.1	1314
50-8-221	35471	36247	5467†	36632	5656†	33847	34086.6	1164	33894	34044.6	1138	33847	34081.7	883	33892	34208.0	897	33836	34141.8	1258	**33711**	34120.3	1207
50-8-222	56600	54512	-	57001	13619†	52738	53360.1	928	53051	53521.2	960	52740	53395.5	1122	53281	53579.8	948	**52603**	53360.3	1509	53094	53642.8	804
50-8-231	54080	53303	-	53941	-	52805	53019.7	844	52756	53049.8	1004	**52699**	52913.3	831	52896	53115.6	1041	52791	53058.1	1311	52940	53091.0	843
50-8-232	62858	62385	6017†	67546	2451†	**61568**	61976.8	856	61852	62099.2	724	61606	62009.1	701	61865	62197.0	654	61713	62091.8	722	61598	62105.5	630

† Execution interrupted before 21,600 seconds (time limit) due to memory limit.

E
Metaheuristic

E.1
Results by Instance for the Iterated Greedy Algorithm

In this section, we present the complete results found by the Iterated Greedy Algorithm (Chapter 7) when running for 2500, 4500, and 7000 iterations, considering the whole benchmark of 72 PLSVSP instances (Chapter 4). Table F.3 shows the results in terms of the minimum (min), average (avg), maximum (max), and standard deviation (sd) of the total weighted completion time achieved by each variant, considering the ten independent runs. The table also includes the average computational time ($\overline{time}$). The first two columns indicate the name of the instance and the Iterated Greedy Algorithm's best solution. To save space, we shorten the instance names as described in Appendix B.4.

Table E.1: Iterated Greedy Algorithm's complete results by instance.

Instance	Best	IG-RG-2500					IG-RG-4500					IG-RG-7000				
		min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$
15-4-111	6903	6903	6903.00	6903	0.00	0.44	6903	6903.00	6903	0.00	0.81	6903	6903.00	6903	0.00	1.24
15-4-112	7717	7717	7717.00	7717	0.00	0.61	7717	7717.00	7717	0.00	1.08	7717	7717.00	7717	0.00	1.68
15-4-121	7498	7498	7498.00	7498	0.00	0.55	7498	7498.00	7498	0.00	0.99	7498	7498.00	7498	0.00	1.50
15-4-122	8608	8608	8608.00	8608	0.00	0.57	8608	8608.00	8608	0.00	1.00	8608	8608.00	8608	0.00	1.53
15-4-131	10964	10964	10964.00	10964	0.00	0.51	10964	10964.00	10964	0.00	0.91	10964	10964.00	10964	0.00	1.40

Continued on next page

Table E.1 – continued from previous page

Instance	Best	IG-RG-2500					IG-RG-4500					IG-RG-7000				
		min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$
15-4-132	10685	10685	10685.00	10685	0.00	0.61	10685	10685.00	10685	0.00	1.08	10685	10685.00	10685	0.00	1.68
15-4-211	8822	8822	8822.00	8822	0.00	0.50	8822	8822.00	8822	0.00	0.89	8822	8822.00	8822	0.00	1.36
15-4-212	6579	6579	6579.00	6579	0.00	0.61	6579	6579.00	6579	0.00	1.07	6579	6579.00	6579	0.00	1.67
15-4-221	5929	5929	5929.00	5929	0.00	0.53	5929	5929.00	5929	0.00	0.93	5929	5929.00	5929	0.00	1.44
15-4-222	14545	14545	14545.00	14545	0.00	0.55	14545	14545.00	14545	0.00	1.00	14545	14545.00	14545	0.00	1.54
15-4-231	10755	10755	10755.00	10755	0.00	0.58	10755	10755.00	10755	0.00	1.02	10755	10755.00	10755	0.00	1.62
15-4-232	15747	15747	15747.00	15747	0.00	0.55	15747	15747.00	15747	0.00	0.98	15747	15747.00	15747	0.00	1.50
15-8-111	4306	4306	4306.00	4306	0.00	0.55	4306	4306.00	4306	0.00	0.97	4306	4306.00	4306	0.00	1.47
15-8-112	5453	5453	5453.00	5453	0.00	0.61	5453	5453.00	5453	0.00	1.05	5453	5453.00	5453	0.00	1.61
15-8-121	10790	10790	10790.00	10790	0.00	0.40	10790	10790.00	10790	0.00	0.71	10790	10790.00	10790	0.00	1.08
15-8-122	8369	8369	8369.00	8369	0.00	0.44	8369	8369.00	8369	0.00	0.78	8369	8369.00	8369	0.00	1.24
15-8-131	4339	4339	4339.00	4339	0.00	0.57	4339	4339.00	4339	0.00	1.01	4339	4339.00	4339	0.00	1.57
15-8-132	7371	7371	7371.00	7371	0.00	0.59	7371	7371.00	7371	0.00	1.08	7371	7371.00	7371	0.00	1.65
15-8-211	3189	3189	3189.00	3189	0.00	0.56	3189	3189.00	3189	0.00	0.98	3189	3189.00	3189	0.00	1.49
15-8-212	4256	4256	4256.00	4256	0.00	0.60	4256	4256.00	4256	0.00	1.07	4256	4256.00	4256	0.00	1.61
15-8-221	5519	5519	5525.60	5555	13.99	0.57	5519	5529.20	5555	16.50	1.00	5519	5525.60	5555	13.99	1.56
15-8-222	10461	10461	10461.00	10461	0.00	0.55	10461	10461.00	10461	0.00	0.97	10461	10461.00	10461	0.00	1.48
15-8-231	8002	8002	8002.00	8002	0.00	0.49	8002	8002.00	8002	0.00	0.85	8002	8002.00	8002	0.00	1.32
15-8-232	5127	5127	5127.00	5127	0.00	0.64	5127	5127.00	5127	0.00	1.14	5127	5127.00	5127	0.00	1.74
25-4-111	9151	9151	9151.00	9151	0.00	2.04	9151	9151.00	9151	0.00	3.66	9151	9151.00	9151	0.00	5.70
25-4-112	18678	18678	18678.00	18678	0.00	2.09	18678	18678.00	18678	0.00	3.73	18678	18678.00	18678	0.00	5.85
25-4-121	22865	22865	22865.00	22865	0.00	2.32	22865	22865.00	22865	0.00	4.15	22865	22865.00	22865	0.00	6.44
25-4-122	12415	12415	12415.00	12415	0.00	2.10	12415	12415.00	12415	0.00	3.78	12415	12415.00	12415	0.00	5.84
25-4-131	32800	32800	32838.40	32854	20.95	2.15	32800	32834.40	32854	25.95	3.84	32800	32819.90	32854	26.16	5.91
25-4-132	27555	27555	27555.00	27555	0.00	1.91	27555	27555.00	27555	0.00	3.35	27555	27555.00	27555	0.00	5.19
25-4-211	29679	29679	29679.00	29679	0.00	1.71	29679	29679.00	29679	0.00	3.05	29679	29679.00	29679	0.00	4.75
25-4-212	19595	19595	19595.00	19595	0.00	1.99	19595	19595.00	19595	0.00	3.50	19595	19595.00	19595	0.00	5.46
25-4-221	19833	19833	19833.00	19833	0.00	1.92	19833	19833.00	19833	0.00	3.41	19833	19833.00	19833	0.00	5.27
25-4-222	27148	27148	27148.00	27148	0.00	2.22	27148	27148.00	27148	0.00	4.03	27148	27148.00	27148	0.00	6.16
25-4-231	29552	29552	29552.00	29552	0.00	1.86	29552	29552.00	29552	0.00	3.32	29552	29552.00	29552	0.00	5.10
25-4-232	29383	29383	29383.00	29383	0.00	2.12	29383	29383.00	29383	0.00	3.80	29383	29383.00	29383	0.00	5.89
25-8-111	11387	11387	11434.10	11509	41.08	2.44	11387	11411.40	11448	26.49	4.36	11387	11391.30	11430	13.60	6.75
25-8-112	16053	16053	16053.00	16053	0.00	2.55	16053	16053.00	16053	0.00	4.52	16053	16053.00	16053	0.00	7.02
25-8-121	10478	10478	10502.00	10538	30.98	2.40	10478	10502.00	10538	30.98	4.28	10478	10484.00	10538	18.97	6.65

Continued on next page

Table E.1 – continued from previous page

Instance	Best	IG-RG-2500					IG-RG-4500					IG-RG-7000				
		min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$	min	avg	max	sd	$\overline{time}$
25-8-122	19647	19647	19647.00	19647	0.00	2.20	19647	19647.00	19647	0.00	3.91	19647	19647.00	19647	0.00	6.07
25-8-131	9700	9700	9700.00	9700	0.00	2.14	9700	9700.00	9700	0.00	3.82	9700	9700.00	9700	0.00	5.88
25-8-132	17911	17911	17916.20	17943	9.74	2.22	17911	17912.00	17916	2.11	3.96	17911	17911.00	17911	0.00	6.10
25-8-211	8261	8261	8283.80	8294	12.26	2.63	8261	8286.50	8300	9.72	4.69	8261	8281.10	8300	14.36	7.24
25-8-212	13337	13337	13392.10	13486	43.03	2.45	13337	13387.40	13433	22.66	4.35	13337	13377.80	13388	21.50	6.81
25-8-221	10993	10993	10994.10	11004	3.48	2.43	10993	10993.00	10993	0.00	4.40	10993	10993.00	10993	0.00	6.71
25-8-222	13806	13806	13816.30	13884	25.05	2.69	13806	13809.80	13831	8.50	4.85	13806	13806.00	13806	0.00	7.48
25-8-231	9230	9230	9230.00	9230	0.00	2.39	9230	9230.00	9230	0.00	4.24	9230	9230.00	9230	0.00	6.49
25-8-232	16863	16867	16887.30	16896	12.25	2.57	16867	16884.10	16896	13.41	4.59	16863	16870.70	16896	9.80	7.07
50-4-111	60849	60849	60977.60	61160	126.47	20.97	60849	60985.20	61160	145.85	37.56	60849	60878.00	61139	91.71	57.94
50-4-112	97260	97260	97535.30	97860	237.62	23.42	97260	97533.90	98058	266.85	42.01	97260	97466.30	97704	218.47	65.27
50-4-121	63251	63251	63423.10	63621	133.27	20.01	63251	63367.10	63620	130.72	36.06	63251	63316.00	63419	70.51	56.07
50-4-122	105884	105974	106168.50	106468	160.86	25.02	105974	106102.50	106229	110.80	45.70	105884	105945.30	106094	65.89	70.07
50-4-131	91144	91144	91188.20	91234	37.20	21.59	91144	91144.80	91152	2.53	39.20	91144	91144.00	91144	0.00	61.10
50-4-132	97889	97941	98104.10	98197	80.43	23.47	97915	98018.80	98248	133.22	41.61	97889	98001.40	98244	119.53	65.01
50-4-211	78714	78716	78999.90	79472	306.17	23.42	78714	78915.50	79405	270.30	42.18	78716	78837.50	79236	198.26	65.24
50-4-212	80798	80798	80947.90	81256	177.59	19.81	80798	80872.60	81203	138.48	35.68	80798	80811.30	80819	9.59	55.58
50-4-221	103316	103345	103349.20	103387	13.28	21.96	103316	103342.10	103345	9.17	39.78	103345	103345.00	103345	0.00	62.63
50-4-222	100694	100694	101140.30	101706	356.45	21.43	100694	100818.40	101536	258.86	38.63	100694	100764.70	101234	168.87	60.43
50-4-231	85140	85219	85319.40	85576	131.55	19.65	85219	85241.40	85298	31.71	34.94	85140	85235.40	85330	51.87	54.68
50-4-232	128964	128964	129016.70	129032	22.49	22.35	128971	129000.70	129042	23.29	40.42	128964	128990.80	129032	22.14	62.37
50-8-111	32981	33037	33142.50	33242	60.62	18.56	33046	33119.70	33184	54.28	33.38	32981	33083.20	33180	64.17	51.64
50-8-112	44341	44692	44872.90	45241	184.45	23.38	44436	44701.80	44975	183.88	42.17	44341	44599.20	44885	157.06	66.22
50-8-121	30195	30211	30269.30	30358	40.71	20.98	30195	30233.60	30339	41.66	37.30	30206	30246.10	30281	26.72	58.57
50-8-122	40462	40529	40677.70	40821	93.02	24.38	40462	40626.80	40774	104.55	43.77	40462	40562.50	40794	101.46	67.96
50-8-131	58598	58637	58783.80	58937	101.44	18.17	58598	58713.00	58913	100.23	32.74	58598	58689.70	58837	87.98	50.75
50-8-132	61763	61782	61970.50	62476	232.15	20.75	61763	61896.10	62163	129.67	37.12	61763	61903.90	62133	139.64	56.82
50-8-211	45431	45606	45876.60	46132	167.93	22.62	45603	45808.30	45980	115.87	41.37	45431	45609.00	45976	169.13	63.81
50-8-212	53890	53943	54187.40	54364	133.93	26.14	53985	54198.20	54683	201.90	46.62	53890	54041.70	54302	109.60	72.28
50-8-221	33418	33505	33571.40	33624	43.75	21.50	33465	33521.70	33624	55.82	38.45	33418	33515.70	33612	65.00	59.78
50-8-222	52032	52274	52358.90	52627	102.34	22.12	52032	52308.90	52495	119.14	39.50	52032	52261.00	52429	104.68	60.81
50-8-231	52499	52702	52775.70	52866	50.52	23.27	52504	52640.50	52831	96.27	42.02	52499	52605.40	52726	74.50	65.08
50-8-232	61450	61502	61578.60	61758	80.57	17.08	61487	61536.50	61708	63.28	30.46	61450	61509.50	61591	40.63	47.43

F
Simheuristics

F.1
Results by Instance and Variance Level for the Simheuristics

In this section, we present the complete results of the simheuristics introduce in Chapter 8 when running for the three different variance levels (Low, Medium, and High), considering the complete benchmark of 72 PLSVSP instances (Chapter 4). Table F.1 shows the results of the low variance scenario in terms of the deterministic objective function value (Det.), the expected value of the stochastic objective values (Exp.), the $\text{CVaR}_{95\%}$ of the stochastic objective values (CVaR), and the computational time (Time), considering the best solution among ten independent runs. The first column indicates the name of the instance. Tables F.2 and F.3 depict the same results for the medium and high variance levels, respectively. To save space, we shorten the instance names as described in Appendix B.4.

Table F.1: Simheuristics complete results by instance regarding the low variance level scenario.

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
15-4-111	6903	6967.18	7559.25	0.54	6903	6959.09	7542.08	1.5	6903	6964.87	7532.47	1.52
15-4-112	7717	8029.6	8856.58	0.73	7785	8015.42	8861.99	2.0	7717	8020.85	8810.58	1.99
15-4-121	7498	7690.44	8272.28	0.59	7498	7686.73	8266	1.3	7648	7730.95	8250.22	1.24
15-4-122	8608	8833.09	9580.45	0.62	8608	8828.19	9579.56	1.3	8699	8896.31	9532.38	1.31
15-4-131	10964	11241.1	12395.7	0.57	10964	11221	12462.1	1.7	11232	11385.7	12202.8	1.64

Continued on next page

Table F.1 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
15-4-132	10685	11003.9	11996.2	0.72	10767	10987.8	11948.3	1.6	10779	10995	11914.2	1.60
15-4-211	8822	8991	9755.85	0.56	8822	8987.2	9744.45	1.6	8822	8991.85	9731.96	1.46
15-4-212	6579	6948.11	7743.11	0.73	6593	6940.74	7711.43	2.3	6691	6977.62	7668.52	2.21
15-4-221	5929	6113.36	6624.97	0.57	5929	6063.48	6530.54	1.3	5967	6086.1	6518.03	1.32
15-4-222	14545	14911.9	15925.3	0.66	14545	14828.8	15848.3	1.1	14807	14927	15834.2	1.15
15-4-231	10755	11139.3	12032.3	0.68	10776	11052	11768.5	1.3	10850	11067.6	11715	1.19
15-4-232	15747	16095	17233	0.63	15747	16081	17204.3	1.2	15747	16090.1	17201.6	1.17
15-8-111	4306	4426.07	4846.69	0.63	4309	4402.88	4842.2	1.8	4343	4403.14	4830.96	1.71
15-8-112	5453	5652.26	6213.02	0.68	5471	5627.78	6220.39	2.0	5453	5651.19	6194.82	1.98
15-8-121	10790	10994.3	12481.2	0.51	10790	10813.6	12393	3.2	10790	10951.5	12398.5	2.81
15-8-122	8369	8668.77	9614.99	0.61	8369	8654.76	9601.81	1.8	8456	8725.93	9579.89	1.74
15-8-131	4339	4411.11	4783.07	0.63	4339	4407.08	4766.17	1.4	4339	4427.38	4770.64	1.47
15-8-132	7371	7638.31	8274.83	0.67	7377	7586.93	8222.21	1.6	7377	7595.45	8210.07	1.49
15-8-211	3189	3272.26	3701.48	0.69	3196	3255.7	3693.57	2.7	3203	3259.75	3695.76	2.65
15-8-212	4256	4481.92	5098.71	0.76	4263	4427.12	4986.13	2.6	4263	4434.82	4973.41	2.54
15-8-221	5519	5708.81	6320.27	0.64	5519	5703.61	6307.01	1.7	5555	5715.08	6216.24	1.67
15-8-222	10461	10879.7	11777.5	0.61	10461	10839.5	11701.5	1.4	10461	10837.9	11696.6	1.42
15-8-231	8002	8177.97	8854.58	0.56	8023	8042.52	8781.74	1.5	8017	8073.41	8782.89	1.48
15-8-232	5127	5365.46	5978.59	0.76	5127	5341.84	5981.22	2.3	5127	5357.62	5969.35	2.56
25-4-111	9151	9470.93	10318.5	2.14	9163	9457.02	10278.6	3.5	9163	9470.12	10299.9	3.48
25-4-112	18678	19073.4	20434.1	2.16	18678	19046	20376.3	2.9	18678	19056.4	20347.2	2.92
25-4-121	22865	23781.6	25358.8	2.37	22865	23768.2	25359.1	3.1	22865	23794.2	25309.5	3.05
25-4-122	12415	12767.6	13691.2	2.21	12415	12731.9	13646.1	3.0	12415	12745.1	13623.4	3.08
25-4-131	32800	33912.8	35864.2	2.15	32907	33483.5	35181.5	2.7	32907	33523.1	35174.4	2.63
25-4-132	27555	28566.7	30406.8	1.96	27555	28518.4	30330	2.6	27816	28606.5	30218.5	2.61
25-4-211	29679	30431.4	32413.8	1.79	29679	30383.6	32273.7	2.4	29679	30430.8	32297.9	2.45
25-4-212	19595	20158.9	21655.4	2.06	19595	20147.8	21656.9	3.0	19595	20156.7	21636.2	3.01
25-4-221	19833	20494.1	21853.9	1.99	19833	20447.9	21753.2	2.6	19840	20456.6	21750	2.76
25-4-222	27148	27923	29729.7	2.24	27192	27520.9	29078.9	2.9	27192	27554.1	29047.2	3.03
25-4-231	29552	30494.2	32258.7	1.87	29632	30241.7	31803.5	2.4	29632	30236.2	31764.4	2.40
25-4-232	29383	30586.8	32323.7	2.22	29589	30246	31836.3	2.7	29589	30250.8	31857.9	2.73
25-8-111	11387	11802.8	12750.1	2.49	11387	11758.8	12723.6	3.6	11433	11812	12721.3	3.50
25-8-112	16053	16752.6	18051.2	2.67	16067	16681.7	17925.6	3.5	16068	16704.8	17924.5	3.49
25-8-121	10478	10937.4	11706.3	2.44	10593	10897.3	11743	3.4	10593	10917.9	11662.7	3.38

Continued on next page

Table F.1 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
25-8-122	19647	20372.2	21612.7	2.39	19752	20217.1	21317.5	2.8	19806	20247.3	21258.2	2.86
25-8-131	9700	10022	10732.2	2.26	9751	9955.83	10657.2	2.9	9728	10040	10636.8	2.96
25-8-132	17911	18891.9	20204	2.26	17966	18701	20040.2	3.2	17966	18705.3	20063.8	3.18
25-8-211	8261	8736.56	9506.17	2.69	8357	8606.38	9443.22	4.1	8443	8692.47	9432.57	4.07
25-8-212	13337	14089.6	15187	2.50	13383	14088.7	15194	3.3	13789	14163.4	15091	3.39
25-8-221	10993	11413.2	12235.6	2.48	10993	11377.3	12179.2	3.3	10993	11388.5	12162.3	3.28
25-8-222	13806	14433.1	15421.3	2.80	13806	14408.9	15368.3	3.5	13918	14441.7	15368.2	3.50
25-8-231	9230	9647.28	10255.7	2.37	9271	9582.43	10205.5	3.1	9271	9591.58	10168.7	3.00
25-8-232	16867	17591.1	18775.1	2.71	16947	17464.9	18624.6	3.4	16957	17473.1	18643.2	3.31
50-4-111	60849	62797.7	66393	20.78	60849	62727	66228	22.0	61238	62893.7	66152.9	21.90
50-4-112	97260	99079.2	103482	23.65	97260	99071.6	103400	24.3	97510	99077.5	103233	24.55
50-4-121	63251	65499.7	68664.5	20.39	63264	65409.1	68416.7	21.2	63720	65691.9	68408.3	21.82
50-4-122	105974	109849	114708	25.72	106209	109491	115076	26.3	106446	109592	114293	26.28
50-4-131	91144	94182.2	98321.4	21.69	91377	93617.3	97177.6	22.7	91377	93661.1	97219.3	23.17
50-4-132	97941	101286	105364	23.73	98102	100869	105092	24.8	98102	100884	104945	24.84
50-4-211	78716	80477.2	84596.8	22.95	78946	80422.4	84142.9	25.3	78716	80494	84316.3	24.22
50-4-212	80798	83418.6	87934.7	20.37	81001	82999.7	87360.3	21.2	81895	83189.7	87059.1	20.86
50-4-221	103345	107609	112093	22.15	103415	107029	111464	23.3	103420	107204	111551	23.32
50-4-222	100694	104056	108682	21.65	100893	103981	108638	22.7	101427	104094	108180	22.42
50-4-231	85219	88778	92256.5	19.89	85684	88689.3	92247.1	20.4	85526	88832.7	92166.8	20.23
50-4-232	128964	131835	136670	22.98	129365	131622	136209	23.0	129365	131686	136222	23.21
50-8-111	33037	34579.9	36590.9	18.77	33348	34424.7	36451.2	20.1	33149	34509.5	36326.9	20.05
50-8-112	44692	46542.2	48931.7	24.31	44716	46524.9	48893	24.9	44807	46617.5	48894	25.05
50-8-121	30211	31665.6	33503.9	20.54	30427	31619.2	33406.5	23.3	30427	31602.8	33271.4	22.23
50-8-122	40529	42638.4	45101.8	24.87	40699	42307	44346.1	25.2	40699	42342.4	44268.3	25.23
50-8-131	58637	60948.4	63430.9	18.22	58671	60895.9	63264.5	19.1	58671	60879.9	63271.3	19.14
50-8-132	61782	64709.1	67853.4	20.42	61891	64128.2	67347	21.8	61891	64168.4	67239.1	21.60
50-8-211	45606	47561.8	50196.8	22.92	45875	47194.8	49630.3	24.3	46000	47198.7	49532.7	23.55
50-8-212	53943	56393.9	59260.7	25.82	54125	56139.7	58841.2	26.9	54239	56270.7	58813.3	27.45
50-8-221	33505	35096.4	36869.7	21.09	33840	34992	36557.8	22.6	33840	35014.9	36558.6	22.28
50-8-222	52274	54115.2	56869.3	22.36	52314	53982.2	56414.2	23.2	52314	54013.2	56364	22.66
50-8-231	52702	55146.2	57699.4	24.24	52956	54741	57076.6	24.7	52956	54742.1	57076.3	24.55
50-8-232	61502	63340.5	66453.1	17.59	61523	63028	66148.5	18.4	61561	63170.2	66148.9	18.11

Table F.2: Simheuristics complete results by instance regarding the medium variance level scenario.

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
15-4-111	6903	7200.89	8624.74	0.69	6903	7190.8	8604.16	5.5	6903	7193.55	8586.43	5.484
15-4-112	7717	8557.67	10634.1	0.96	7834	8489.71	10542.6	7.8	7840	8503.12	10521.5	7.407
15-4-121	7498	8008.27	9465.75	0.78	7648	7968.77	9265.84	4.2	7648	7989.65	9281.57	4.081
15-4-122	8608	9338.07	11160.7	0.78	8732	9328.65	11042.7	4.8	8756	9341.77	10965.8	4.701
15-4-131	10964	11749.7	14152.2	0.74	11084	11656.5	14239.3	5.2	11432	11738	13508.9	5.042
15-4-132	10685	11556.7	13726.1	0.82	10779	11458.2	13449.9	4.8	10779	11460	13433.9	4.712
15-4-211	8822	9357.56	11177.1	0.73	8947	9279.73	11109.5	5.2	8822	9356.23	11148.1	5.132
15-4-212	6579	7517	9419.67	1.03	6828	7415.91	9076.97	7.8	6828	7436.94	9094.42	7.692
15-4-221	5929	6486.33	7746.33	0.73	5967	6375.76	7474.51	4.6	5967	6377.57	7485.83	4.525
15-4-222	14545	15563.7	18114.3	0.74	14807	15455.4	17806.2	3.5	14807	15462.5	17759.6	3.467
15-4-231	10755	11842.4	13940.8	0.82	10776	11588.2	13375.4	3.8	10904	11609.8	13313.1	3.922
15-4-232	15747	16761.5	19420.2	0.75	15747	16747.7	19430	3.7	16229	16937.9	19221.7	3.421
15-8-111	4306	4656.78	5648.68	0.82	4309	4627.64	5616.15	6.0	4309	4627.26	5612.79	5.871
15-8-112	5453	5936.24	7231.47	0.91	5544	5877.16	7343.79	6.9	5453	5933.95	7204.3	6.559
15-8-121	10790	11277	14476.9	0.92	10790	11002.2	14320.4	10.6	10790	11051.6	14317.8	9.771
15-8-122	8369	9237.29	11263.6	0.74	8456	9149.08	11217.3	6.2	8624	9272.3	11208.5	5.896
15-8-131	4339	4594.97	5462.04	0.84	4339	4587.52	5443.32	4.6	4411	4672.25	5436.32	4.491
15-8-132	7371	8067.59	9498.1	0.81	7432	7940.58	9366.9	4.7	7408	7984.83	9355.5	4.583
15-8-211	3189	3473.77	4443.43	1.05	3196	3427.63	4368.47	9.0	3203	3432.51	4380.11	8.858
15-8-212	4256	4869.79	6257.47	1.04	4263	4773.73	6092.54	8.8	4263	4775.13	6042.34	8.697
15-8-221	5519	6070.19	7608.39	0.91	5585	6040.72	7281.98	5.8	5585	6079.59	7269.69	5.617
15-8-222	10461	11597.2	13765.4	0.77	10615	11526.9	13603.5	4.9	10615	11558.2	13636.5	4.738
15-8-231	8002	8542.12	9982.22	0.69	8035	8267.15	9808.33	4.7	8023	8270.7	9762.43	4.49
15-8-232	5127	5722.46	7109.99	0.98	5166	5695.66	7129.24	7.4	5145	5697.13	7091.42	7.556
25-4-111	9151	9980.06	12043.4	2.44	9332	9933.2	11933.9	8.7	9311	9957.55	11940.8	8.638
25-4-112	18678	19902.5	23119.2	2.30	18678	19878.4	23049.8	6.3	18772	19912.7	23032	6.095
25-4-121	22865	25232.9	28878.9	2.47	22959	25152.6	28551.5	5.8	23132	25163.1	28550.6	5.722
25-4-122	12415	13427.9	15572.4	2.31	12444	13358.1	15498.9	6.5	12697	13515.7	15409.9	6.238
25-4-131	32800	35653.5	40117.9	2.32	32877	34996.6	39120.1	4.5	32907	35035.4	39206.7	4.533
25-4-132	27555	30607.1	34786.2	2.07	27866	30352.8	34146.8	4.9	27866	30334.1	34078.7	4.847
25-4-211	29679	31888	36612.1	1.90	29971	31643.2	36151	5.3	29918	31890.6	36270.2	5.152
25-4-212	19595	21294.8	24776.4	2.28	19595	21273.8	24735.4	6.9	19821	21294.3	24761	6.689
25-4-221	19833	21600.2	24824.2	2.11	19833	21531.8	24764.9	5.3	20036	21555.9	24542.9	5.422

Continued on next page

Table F.2 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
25-4-222	27148	29395.7	33688.2	2.41	27316	28438	32143.1	5.3	27316	28409.3	32020.1	5.354
25-4-231	29552	31958.1	36256.1	2.02	29632	31598.7	35578.6	4.4	29775	31717	35457.8	4.381
25-4-232	29383	32381.6	36344.2	2.30	29589	31839.5	35745.4	4.7	29589	31860.1	35813.5	4.711
25-8-111	11387	12573.6	14926.3	2.69	11442	12520.9	14906.4	7.7	11559	12577.7	14929.8	7.824
25-8-112	16053	17829.1	20921.5	2.86	16213	17572.6	20383.6	6.9	16250	17636.8	20332.1	6.727
25-8-121	10478	11627.2	13483.6	2.60	10593	11513.6	13454.2	6.9	10604	11552.6	13365.7	6.608
25-8-122	19647	21689.9	24750.8	2.50	19752	21328.8	24202.9	5.1	19806	21389.8	24173.1	5.045
25-8-131	9700	10642.3	12397.1	2.35	9748	10507.2	12225.1	6.0	9832	10585.7	12091	5.674
25-8-132	17911	20359	23617.6	2.50	17966	20033.6	23365.2	6.4	18337	20270.8	23257	6.286
25-8-211	8261	9458.12	11357.6	2.94	8354	9202.04	11217.4	9.0	8471	9257.36	11047.8	8.94
25-8-212	13337	15268.7	18023.2	2.79	13673	15024	17465.6	7.0	13673	15049	17481.5	7.059
25-8-221	10993	12229.6	14307.3	2.66	11136	12055.1	14026.7	6.7	11294	12192.2	14004.1	6.719
25-8-222	13806	15583.9	18195.2	2.96	13948	15446.2	17859.8	6.6	13912	15504.2	17947.4	6.631
25-8-231	9230	10223.5	11699.4	2.53	9289	10079.2	11350.9	5.1	9271	10085.9	11401.6	5.232
25-8-232	16867	18689.7	21313.3	2.81	17134	18382.7	21140.3	5.8	16918	18502.4	21021.9	5.705
50-4-111	60849	66243.4	74204.5	20.86	60914	66231.9	74265.1	24.7	60898	66217	74136	24.734
50-4-112	97260	102741	113268	23.62	97982	102237	112951	26.0	97996	102039	111728	25.931
50-4-121	63251	68851.7	76261.5	20.50	64005	68453.6	74688.8	23.3	63936	68513	74911.1	22.964
50-4-122	105974	116016	127581	25.71	106177	115054	126793	28.6	106957	115472	126499	28.641
50-4-131	91144	98655.8	107632	21.69	91363	97768.6	106239	24.5	91419	97883	106124	23.954
50-4-132	97941	106790	116654	23.37	98254	106358	115862	26.3	98270	106549	115827	27.247
50-4-211	78716	83757	93398.5	23.00	78946	83408.7	92575.2	27.8	78783	83586.8	92818.9	26.754
50-4-212	80798	87422.2	97764.6	20.61	81576	85944.6	95607.8	23.1	82057	86505.7	95376.5	23.603
50-4-221	103345	113478	123878	22.26	104411	112561	122384	24.8	104585	112724	122389	24.139
50-4-222	100694	109567	120136	21.88	101427	109003	118771	24.0	101427	109011	118771	24.539
50-4-231	85219	94110.4	102132	19.96	85683	93864.3	101946	22.1	86975	94435	101897	21.711
50-4-232	128964	136876	148638	22.97	130086	136411	147981	25.4	129271	136664	147963	24.795
50-8-111	33037	36789.9	41566.4	19.98	33485	36387.5	40786.7	23.2	34180	36704	40890.8	22.978
50-8-112	44692	49821.1	55957.2	24.15	44968	49690	55485.7	27.9	44860	49844.8	55723.5	27.818
50-8-121	30211	33969.2	38389.8	20.81	30478	33646	37703.7	25.3	30831	33854.5	37675.3	25.207
50-8-122	40529	45880.4	51986.6	25.00	40963	45287.5	50682	28.8	41432	45483.1	50656.2	29.431
50-8-131	58637	64870.3	70825.9	18.17	58797	64760.9	70491.1	20.4	58797	64759.4	70463.1	20.367
50-8-132	61782	69281.8	76964.5	20.30	61901	68291.2	75841.8	23.9	62003	68397.1	75496.1	22.947
50-8-211	45606	50598.5	56952.9	22.69	45929	49637.2	55484.7	26.6	46044	49939.9	55671.5	26.701

Continued on next page

Table F.2 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
50-8-212	53943	60271.8	67323.1	26.09	54778	59528.4	66175.2	30.5	54201	59607.4	66207.2	30.536
50-8-221	33505	37719.7	41995.5	21.19	33840	37284.7	41196.8	24.5	34309	37376.2	41254.6	24.681
50-8-222	52274	57837.4	64608.2	22.05	52469	57399.2	63842.9	26.1	52314	57533.2	63788.3	24.907
50-8-231	52702	58661	64512.8	24.29	53006	57796.1	63165.2	26.2	52972	58068	63730.5	25.907
50-8-232	61502	66672.7	73535.2	17.36	61632	66124.5	73014	19.5	61523	66318.9	73147.9	20.441

Table F.3: Simheuristics complete results by instance regarding the high variance level scenario.

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
15-4-111	6903	7592.13	10391.6	1.18	6903	7579.85	10401.8	15.6	6903	7590.32	10347.5	15.397
15-4-112	7717	9266.46	13388.5	1.68	7990	9121.23	12964	20.1	7990	9124.46	12955.7	19.713
15-4-121	7498	8468.77	11341.7	1.13	7648	8364.63	10898	11.1	7714	8389.04	10764.7	11.002
15-4-122	8608	10032.9	13527.9	1.18	8732	9942.8	13283.6	13.2	9133	10153.8	13152.3	12.666
15-4-131	10964	12483.5	16704.6	1.10	11432	12255.4	15685.5	12.3	11432	12255.3	15659.4	11.402
15-4-132	10685	12369.9	16259.3	1.16	10849	12181.8	15803.8	11.4	10849	12185.3	15784.7	11.366
15-4-211	8822	9929.15	13439.1	1.21	8947	9704.03	13146.5	14.0	8947	9731.26	13190.4	14.689
15-4-212	6579	8225.96	11961.3	1.69	6943	8002.88	11235.6	20.2	6894	8052.82	11247.3	20.164
15-4-221	5929	6973.7	9341.17	1.10	5967	6794.16	8924.88	12.0	6225	6938.31	8897.91	12.007
15-4-222	14545	16507.5	21496.6	1.06	14807	16368.3	21071.2	9.8	14934	16385	20694.3	9.569
15-4-231	10755	12792.1	16744	1.08	10904	12292.6	15555.4	9.8	10904	12289.5	15547.2	9.889
15-4-232	15747	17714.5	22691.1	0.99	15747	17718.4	22678.6	9.0	16229	17753	22127	9.038
15-8-111	4306	4974.92	6896.29	1.37	4320	4958.05	6861.37	16.3	4335	4972.94	6863.14	16.095
15-8-112	5453	6313.6	8765.25	1.45	5553	6191.71	8622.97	17.5	5553	6199.09	8624.07	17.205
15-8-121	10790	11755.2	17279.5	1.51	10790	11470.8	17062	24.1	10790	11487.9	17032.6	22.714
15-8-122	8369	10037	13841.2	1.22	8539	9861.58	13672.1	16.1	8539	9918.18	13702.2	16.04
15-8-131	4339	4869.54	6464.89	1.17	4477	4860.17	6488.57	12.1	4411	4944.56	6380.07	11.659
15-8-132	7371	8647.1	11351.4	1.22	7456	8446.13	11140.1	12.8	7382	8496.8	11098.5	11.24
15-8-211	3189	3772.19	5644.29	1.80	3250	3686.54	5406.79	23.3	3250	3686.32	5409.63	22.916
15-8-212	4256	5364.58	8046.73	1.85	4296	5235.95	7776.85	23.4	4263	5243.53	7771.39	23.312

Continued on next page

Table F.3 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
15-8-221	5519	6572.06	9514.02	1.55	5593	6510.18	8932.09	15.0	5649	6568.97	8870.1	15.041
15-8-222	10461	12541.6	16783.3	1.17	10615	12428.1	16530.7	13.0	10615	12471.3	16590.1	13.019
15-8-231	8002	9055.71	11786.9	0.98	8035	8697.39	11424.5	11.4	8023	8704.63	11333.7	11.147
15-8-232	5127	6228.23	8850.86	1.51	5166	6191.38	8861.82	19.4	5166	6210.21	8806.29	18.936
25-4-111	9151	10716.1	14727.9	3.06	9332	10604.6	14431.2	20.9	9584	10760.1	14299.3	20.725
25-4-112	18678	21213.6	27447	2.70	18772	21153.7	27319.3	14.1	19212	21327.9	27106.1	13.915
25-4-121	22865	27162	34061.3	2.82	23287	26874.4	33328.6	11.7	23268	27043.7	33185.9	11.727
25-4-122	12415	14367.9	18496.9	2.71	12444	14243	18300	14.3	12709	14465.1	18184.7	13.895
25-4-131	32800	38101.9	46318.3	2.47	32877	37279.9	45182.9	8.9	33619	37659.5	44860.8	8.82
25-4-132	27555	33438	41250.3	2.28	28068	32735.3	40003.4	10.2	27899	32740.2	39939.1	10.012
25-4-211	29679	33917.5	42916	2.24	29971	33528.9	42534	12.2	29902	33728.5	42198.5	11.751
25-4-212	19595	22901	29604.5	2.62	19821	22808.7	29515.9	15.2	20232	22835.6	29267.3	15.175
25-4-221	19833	23188.6	29205.4	2.39	20036	23109.6	29131.5	11.6	20453	23346.1	28546.9	11.406
25-4-222	27148	31533.9	39736.1	2.71	27316	30023.4	37421.8	11.7	27352	30422.5	37336.4	11.491
25-4-231	29552	34086.4	42330.3	2.27	29852	33625	40638.9	9.5	30642	33921.7	40687.7	9.349
25-4-232	29383	34830.9	42243.4	2.50	29540	34144.6	41624.9	9.3	29449	34400.8	41615.9	9.428
25-8-111	11387	13689.2	18476.9	3.29	11559	13569.3	18425.8	18.7	11779	13834.3	18324.5	18.312
25-8-112	16053	19320.4	25444.4	3.33	16187	18880.6	24367.9	15.3	16187	18880.6	24367.9	15.45
25-8-121	10478	12525.3	16090.7	3.05	10593	12376.3	16028.8	14.2	10661	12406.4	15763.7	13.734
25-8-122	19647	23504.4	29680.7	2.83	19992	22962.2	28784.5	11.6	20428	23165.8	28749.5	11.03
25-8-131	9700	11510.1	14918.3	2.77	9856	11296.8	14407.4	13.6	10155	11529.8	14226.5	13.307
25-8-132	17911	22358.5	28826	2.79	17966	22012.6	28605.3	14.6	18306	22068.1	27890	14.286
25-8-211	8261	10392.3	14095.3	3.52	8451	9970.56	13505.7	21.6	8425	10005.1	13540.5	20.59
25-8-212	13337	16845.7	22490.4	3.24	13673	16429.4	21570.5	16.4	13658	16450	21621.7	16.356
25-8-221	10993	13346.1	17466.2	3.10	11136	13070.8	16965.1	14.9	11425	13258.7	16752.8	14.441
25-8-222	13806	17199.8	22593.7	3.41	13948	16911.6	21894.4	14.9	13948	16905	21859.1	15.017
25-8-231	9230	10990.8	13739.5	2.86	9494	10729	13307	10.4	9360	10802.8	13195	10.428
25-8-232	16867	20302	25318.8	3.06	17134	19749.5	24892.1	11.4	16918	19930.6	24686.1	11.543
50-4-111	60849	71340.7	86574.9	21.10	61024	71202.6	86534.6	31.7	60909	71246.2	86206	32.07
50-4-112	97260	108029	128188	24.15	97996	106866	125970	31.4	98856	107708	126266	31.396
50-4-121	63251	73656.9	87383.6	20.59	64005	72575.5	84413.6	27.7	64005	72564.6	84405.9	27.406
50-4-122	105974	124336	145771	26.13	106684	122479	142613	33.6	107047	122598	142385	32.545
50-4-131	91144	104752	120755	21.83	92519	103000	117300	27.1	93179	104108	117952	27.052
50-4-132	97941	114429	132564	23.72	98960	113786	130465	30.2	98293	113969	131130	30.284

Continued on next page

Table F.3 – continued from previous page

Instance	IG				SimIG-Exp				SimIG-CVaR			
	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time	Det.	Exp.	CVaR	Time
50-4-211	78716	88579.5	106094	23.41	78946	88111	105449	34.0	79958	89002.9	105718	33.309
50-4-212	80798	92861.1	112059	20.73	81576	90840.5	109798	29.4	82612	91277.8	108625	30.38
50-4-221	103345	121364	140427	22.13	105521	119819	137454	29.0	104644	120580	138015	28.934
50-4-222	100694	117098	136249	22.81	102318	115893	135122	28.3	101427	116164	134063	27.892
50-4-231	85219	101200	116080	20.23	85684	100713	115686	25.0	86678	101276	114738	25.415
50-4-232	128964	144538	166641	23.40	129933	143579	165577	28.6	131915	143859	163640	28.631
50-8-111	33037	39826.7	49171.3	19.40	33485	39072.1	47742.6	31.4	34180	39269.2	47430.7	31.123
50-8-112	44692	54532.5	67449.8	24.63	44968	54222.8	66445	36.0	46425	54618.7	66472	35.964
50-8-121	30211	37174.1	45567.2	21.04	30970	36551.6	43945.6	32.1	30935	36533.9	43859.8	33.043
50-8-122	40529	50512.1	62700.7	26.26	41040	49643.8	60403.7	37.7	40737	49881.5	60603.4	36.226
50-8-131	58637	70363	81846.6	18.42	59000	69963.2	81320.7	24.5	58797	70137.7	81102.4	24.161
50-8-132	61782	75857.2	91045.5	20.69	62925	74071.3	88008.8	30.0	64753	74964.3	88196.1	31.69
50-8-211	45606	54693.4	67294.6	23.21	46149	53303.6	65142.7	34.7	45872	53213.9	64925.8	34.287
50-8-212	53943	65510.8	79795.7	26.45	54778	64357	78216.6	37.7	54201	64501.9	78049	37.335
50-8-221	33505	41298.2	49788.7	21.79	34309	40557.2	48141.6	30.4	34462	40752.8	48082	30.777
50-8-222	52274	63398.9	77134.1	22.35	52643	62735.5	75754.3	32.0	53477	63677.8	75853	31.639
50-8-231	52702	63532.3	74542.7	24.45	52956	62460.8	73032.2	30.9	53682	62498.9	72665.1	30.341
50-8-232	61502	72003.2	85085.1	17.52	61583	71344.7	84537.2	25.4	62340	71625.3	83841.6	25.213