UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Gabriela Braga Fonseca

ABORDAGENS ROBUSTAS PARA O PROBLEMA DE SEQUENCIAMENTO FLOW SHOP CROSS-DOCKING

Belo Horizonte
2021

Gabriela Braga Fonseca

ABORDAGENS ROBUSTAS PARA O PROBLEMA DE SEQUENCIAMENTO FLOW SHOP CROSS-DOCKING

Tese apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Doutora em Engenharia de Produção.

Área de concentração: Pesquisa Operacional e Intervenção em Sistemas Sociotécnicos.

Linha de pesquisa: Modelos e Algoritmos de Otimização para Sistemas em Redes e de Produção.

Orientador: Martín Gómez Ravetti
Coorientador: Thiago Henrique Nogueira

Belo Horizonte
2021

Fonseca, Gabriela Braga.
F676r Abordagens robustas para o problema de sequenciamento flow shop cross-docking [recurso eletrônico] / Gabriela Braga Fonseca. - 2021.
1 recurso online (ix, 79 f. : il., color.) : pdf.

Orientador: Martín Gómez Ravetti.
Coorientador: Thiago Henrique Nogueira.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 58-79.

Bibliografia: f. 53-57.

1. Engenharia de produção - Teses. 2. Incerteza - Teses. 3. Logística - Teses. I. Gómez Ravetti, Martín. II. Nogueira, Thiago Henrique. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 658.5(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo Cesar Vital dos Santos - CRB-6/2165
Biblioteca Prof. Mário Werneck - Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
Escola de Engenharia
Programa de Pós-Graduação em Engenharia de Produção

FOLHA DE APROVAÇÃO

ABORDAGENS ROBUSTAS PARA O PROBLEMA DE SEQUENCIAMENTO FLOW SHOP CROSS-DOCKING

GABRIELA BRAGA FONSECA

Tese submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA DE PRODUÇÃO, como requisito para obtenção do grau de Doutor em ENGENHARIA DE PRODUÇÃO, área de concentração PESQUISA OPERACIONAL E INTERVENÇÃO EM SISTEMAS SOCIOTÉCNICOS, linha de pesquisa Mod. e Algorit. de Otimiz. para Sistemas em Redes e de Prod..

Aprovada em 09 de julho de 2021, pela banca constituída pelos membros:

Prof(a). Martin Gomez Ravetti - Orientador
UFMG

Prof(a). Reinaldo Morabito Neto
UFSCAR

Prof(a). Geraldo Robson Mateus
UFMG

Prof(a). Felipe Campelo Franca Pinto
UFMG

Prof(a). Eduardo Gontijo Carrano
UFMG

Prof(a). Thiago Henrique Nogueira
UFV

Belo Horizonte, 09 de julho de 2021.

Documento assinado eletronicamente por **Martin Gomez Ravetti**, **Professor do Magistério Superior**, em 09/07/2021, às 14:30, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Eduardo Gontijo Carrano**, **Subchefe de departamento**, em 09/07/2021, às 18:28, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Felipe Campelo Franca Pinto**, **Professor do Magistério Superior**, em 12/07/2021, às 08:04, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Geraldo Robson Mateus**, **Professor Magistério Superior - Voluntário**, em 12/07/2021, às 11:06, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Thiago Henrique Nogueira**, **Usuário Externo**, em 15/07/2021, às 06:59, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Reinaldoi Morabito Neto**, **Usuário Externo**, em 15/07/2021, às 10:03, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0828216** e o código CRC **9A841E79**.

Referência: Processo nº 23072.235144/2021-70 SEI nº 0828216

Acknowledgements

I thank God for giving me the ability and wisdom necessary to achieve such an achievement. My eternal gratitude to my advisors, Professor Dr. Martín Gómez Ravetti and Dr. Thiago Henrique Nogueira, for all their support and guidance during my years as a Ph.D. student at UFMG. Special thanks to my family; without their help, I couldn't make it this far.

This research was partially funded by FAPEMIG, CAPES and CNPq, Brazil.

Resumo

A globalização e o rápido crescimento do *e-commerce* têm ajudado na mudança dos hábitos de compra de bens e serviços, aumentando a complexidade dos ambientes logísticos. *Cross-docking* é uma solucão logística com o objetivo de mover produtos diretamente de diferentes fornecedores ou fabricantes e consolidá-los em destinos de entrega final comuns sem armazenamento de longo prazo. Essa estratégia permite agilidade nas entregas e redução dos custos de armazenagem e transporte, além de elevar o nível do serviço logístico. O sucesso da estratégia depende de uma operação de transbordo eficiente. Este trabalho realiza um estudo de sequenciamento de caminhões em um centro de *cross-docking*. Os objetivos específicos envolvem o desenvolvimento de modelos, algoritmos e métodos de resolução para o problema de sequenciamento de caminhões em centros de cross-docking, adicionando aspectos de incerteza. O problema é inicialmente modelado como um problema de sequenciamento *flow shop* de duas máquinas com restrições de precedência, com o objetivo de minimizar o *makespan*, e posteriormente generalizado para o caso de múltiplas docas paralelas. Propomos um método híbrido baseado em uma técnica de relaxação Lagrangiana por meio do algoritmo do volume. Usando informações dos multiplicadores de Lagrange, heurísticas construtivas com procedimentos de busca local geraram boas soluções viáveis. Através de uma série de cortes, a metodologia encontra limites estreitos para tamanhos de instância pequenos e grandes, superando os resultados atuais. Para aproximar nossa abordagem da operação real de *cross-docking*, incorporamos incerteza na data de chegada dos caminhões. Uma abordagem de resequenciamento é fornecida, e um novo algoritmo para resolver o problema de sequenciamento de caminhões com múltiplas docas sob incerteza na data de chegada do caminhão é proposto. Estudamos dois problemas de otimização, o problema de minimizar o *Makespan* (Problema C_{max}) e o problema de minimizar o Tempo Total de Conclusão Ponderado (Problema WC). Comparamos três metodologias e mostramos que o método de resequenciamento pode apoiar os gestores em suas operações diárias de *cross-docking*, lidando de forma eficiente com dados dinâmicos e incertos, levando a boas decisões muito rapidamente.

Palavras-chave: Logística. Sequenciamento. *Cross-docking*. Incertezas.

Abstract

Globalization and the rapid growth of e-commerce have helped change purchase goods and service habits, increasing the complexity of logistics environments. Cross-docking is a logistics solution aiming to move products directly from different suppliers or manufacturers and consolidate them to common final delivery destinations without long-term storage. This strategy allows for fast deliveries and reduced warehousing and transportation costs while increasing logistical service. The success of the strategy depends on an efficient transshipment operation. This work undertakes a study of truck scheduling in a cross-docking center. The specific goals involve developing models, algorithms, and resolution methods for the truck scheduling problem in cross-docking centers, adding aspects of uncertainty. The problem is first modeled as a two-machine flow shop scheduling problem with precedence constraints to minimize the makespan and later generalized it to the parallel-dock case. We propose a hybrid method based on a Lagrangian relaxation technique through the volume algorithm. Using information from the Lagrangian multipliers, constructive heuristics with local search procedures generate good feasible solutions. With a series of cuts, the methodology finds tight bounds for small and large instance sizes, outperforming current results. To approximate our approach to the real cross-docking operation, we incorporate uncertainty in truck arrival times. A rescheduling approach is provided, and a novel algorithm for solving multi-dock truck scheduling problems is proposed under truck arrival time uncertainty. We discuss two optimization problems, the problem of minimizing the Makespan (Problem C_{max}) and the problem of minimizing the Total Weighted Completion Time (Problem WC). Extensive experimentation allows us to compare three methodologies and show that the rescheduling methodology can support managers in their daily cross-docking operations, efficiently handling dynamic and uncertain data, making good decisions quickly.

Keywords: Logistics. Scheduling. Cross-docking. Uncertainties.

List of Figures

List of Tables

List of Algorithms

Contents

Chapter 1

Introduction

Scheduling is a decision-making process that is very important in various manufacturing and services industries. It deals with assigning a set of jobs to a set of machines in a reasonable amount of time to optimize one or more objectives. Scheduling plays an essential role in most manufacturing and production systems and most information processing environments. It is also crucial in transportation and distribution settings and in other types of service industries (Pinedo (2008)).

Scheduling problems can be found in many areas, from production planning to biochemistry problems. Scheduling problems are classified into different types of problems. One of the scheduling problems that has been extensively studied over the last decades is the Flow Shop Scheduling Problem (FSP).

The Flow Shop Scheduling Problem (FSP) is a well-known scheduling problem that can be described as follows: there are m machines in series. A set J of n jobs has to be processed by a set M of m machines. All jobs have to follow the same operation sequence, i.e., they have to be processed first on machine 1, then on machine 2, and so on. In the FSP, the goal is to find a sequence of jobs so that a given criterion is optimized.

The FSP has applications in different industry sectors: metallurgical, chemical, textile, steel, etc. More recently, the flow shop scheduling problem is being applied to effective supply chain management and logistics, especially in cross-docking centers.

The actual market environment, characterized by competition, the globalization of the economy, and an accelerated technological revolution, has led companies to improve their production, logistics, and distribution systems. Customers demand better and faster services, direct-to-customers strategies, and many other new requirements that expect more efficient and integrated logistic operations. Cross-docking is one logistic technique that can reduce inventory costs while increasing the goods flow, improving the efficiency of the supply chain.

Cross-docking (CD) is widespread throughout the world. Several well-known companies such as retail chains (WalMart, Stalk et al. (1991)), mailing companies (UPS, Forger (1995)), automobile manufacturers (Toyota, Witt (1998)), and logistics providers (Gue (1999), Kim et al. (2008)) have gained considerable competitive advantages by using CD.

The main idea behind cross-docking centers is to receive products from different suppliers or manufacturers and consolidate them to common final delivery destinations. Compared to traditional warehouses, a Cross-docking Distribution Center (CDC) is managed with minimal handling and little or no storage between the unloading and loading of goods. This practice can serve different goals, the consolidation of shipments, a shorter delivery lead time, and reducing costs. Readers are referred to Ladier and Alpan (2016a), for a survey discussing industry practices and CDC problem characterization.

Cross-docks raise numerous optimization questions, either strategic, tactical or operational. In a CDC, scheduling decisions are particularly relevant to ensure a rapid turnover and on-time deliveries. Due to its real-world importance, several truck-scheduling works and procedures have been introduced during recent years, treating specific cross-dock settings.

A schematic representation of processes at a CDC is illustrated in Figure 1.1. Firstly, incoming trucks arrive at the center's yard; if the number of trucks is higher than the number of docks, some of them have to wait in a queue until further assignment. Secondly, after being docked, goods of the inbound trucks are unloaded, scanned, sorted, moved across the dock, and loaded into outbound trucks for immediate delivery elsewhere in the distribution chain. Once an outbound (inbound) truck is completely loaded (unloaded), the truck is removed from the dock, replaced by another truck, and the course of action repeats.

Five operations are usually carried out in a conventional distribution center when managing products: receiving, sorting, storing, picking, and shipping operations. All of which can be done consecutively or in a particular order. As a result of applying a cross-docking system, can achieve a reduction of the cost of both storage and products picking by synchronizing the inbound and outbound trucks flows. There are many benefits with the cross-docking: costs reduction (warehousing, inventory-holding, handling, and labor costs), shorter delivery lead times (from supplier to customer, an increase of throughput), improved customer service and customer satisfaction, reduction of storage space, faster inventory turnover, fewer overstocks, reduced risk of loss and damage, etc. However, efficient transshipment processes and careful operations planning become indispensable within a CDC. Flows need to be synchronized to maintain the stock level as low as possible and ensure online deliveries. On the other hand, a reduced stock

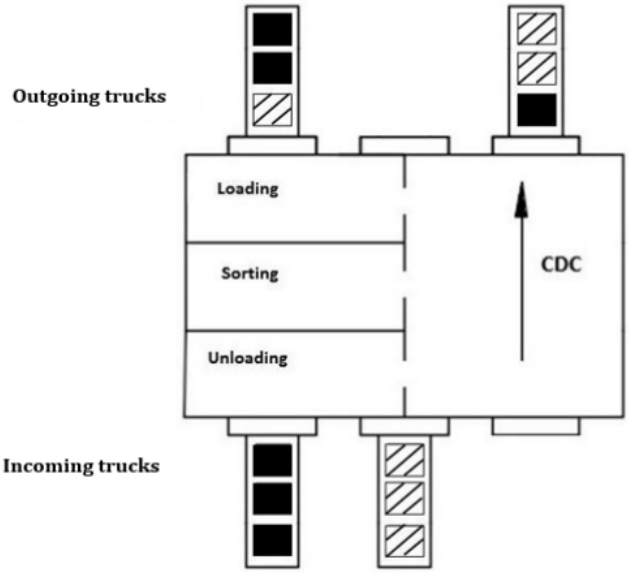


Figure 1.1: Schematic representation of a CDC.

increases the risk of shortage under perturbations or disruptions, as we live in many sectors. Many articles in the literature develop computerized scheduling procedures, which have achieved good results (See examples in Table 2.1).

Our research considers a time-indexed integer programming formulation for a flow shop scheduling problem with cross-docking precedence constraints. We first consider the 2-dock case denoted as $F2|CD|C_{max}$, and strongly NP-hard (See Chen and Lee (2009)), and later we deal with the parallel dock generalization, $F2(P)\ |CD|C_{max}$. Computational experiments show that the model is efficient when solving small problems.

A Hybrid Lagrangian Metaheuristic approach is also proposed and tested. The work has a similar line of thought than Boschetti and Maniezzo (2009). However, it emphasizes the use of the information of the Lagrangian Multipliers as discussed in Pirkwieser et al. (2007), and Paula et al. (2010). The method can efficiently generate strong lower and upper bounds. Results are compared to the best heuristics in the literature introduced by Chen and Lee (2009) (based on Johnson's algorithm), with the CDH heuristic developed by Cota et al. (2016) for the scenario of a parallel machine, and to the best heuristics proposed by Chen and Song (2009) for the multiple parallel processors (multiple docks) case. Better solutions are consistently obtained for all instances tested.

For the 2-dock case, although the subproblems of the Lagrangian relaxation are polynomial, we improve the linear lower bounds through a series of cuts on the makespan value. The proposed algorithm uses the information obtained from the Lagrangian mul-

tipliers to construct feasible solutions through an NEH-based heuristic (Nawaz et al. (1983)) and performs search procedure through a Local Search framework. Furthermore, the proposed hybrid algorithm proves optimality in several instances.

We generalize our approach to the parallel-dock case. In this scenario, the subproblems are NP-hard. Focusing on improving the feasible solutions, we solve the subproblem's linear relaxations to feed the Lagrangian Metaheuristic. Results show that the technique outperforms current heuristic methods.

To approximate our approach to the real cross-docking operation, we incorporate uncertainty in truck arrival times. An efficient rescheduling approach and a novel algorithm for solving multi-dock truck scheduling problems are proposed under truck arrival time uncertainty. Two optimization problems, the problem of minimizing the Makespan (Problem C_{max}) and the problem of minimizing the Total Weighted Completion Time (Problem WC), are presented. Experimental results are discussed by comparing three different methodologies: with adjustments (RA), without adjustments (WA), and perfect information (PI). Results show that our methodology can support managers in their daily cross-docking operations, handle dynamic and uncertain data, and solve online problems.

The text of this work consists of six chapters. The subsequent chapters are organized as follows. An overview of related works is presented in 2. The 2-dock cross-docking flow shop scheduling problem is described in 3, where the mathematical model and the Hybrid Lagrangian Metaheuristic Framework are described, and the experimental results are discussed. Chapter 4 generalizes the algorithm for the parallel machine case. The cross-docking problem is modeled as a hybrid two-stage flow shop scheduling problem with identical machines and cross-docking constraints. The multi-dock truck scheduling problem under truck arrival time uncertainty is studied in Chapter 5. We analyzed two different problems and proposed a novel Rescheduling Approach (RA). To measure the applicability of the proposed methodology, we proceed with the comparison of three different methodologies, enabling a series of analyses that we believe will help CDC managers deal with the uncertainties generated by delays or advances in truck arrivals. Finally, in Chapter 6 conclusions are drawn.

Chapter 2

Literature Review

The purpose of this chapter is to provide a review of the literature on cross-docking. We will highlight some works that deal with cross-docking problems and present recent surveys addressing uncertainties in cross-docking environments.

2.1 Cross-docking

The increasing use of cross-docking techniques has motivated several authors to investigate new ways to improve CDCs' design and tactical operations. In the literature, several decision problems are studied concerning strategic, tactical, and operational decisions.

Boysen and Fliedner (2010), and Belle et al. (2012) focus on the review of the existing literature of cross-docking problems. According to Boysen (2010), can allocate decision problems in a cross-docking center according to the following classification, ordered from strategic to operational levels: location of cross-docking centers, layout, vehicle routing, dock's assignments of destinations, truck scheduling, and resource scheduling inside the center. Here we focus on truck scheduling problems, where the aim is deciding where and when should process the trucks.

Examples of strategic decisions in a CDC can be found in many works. Location of a cross-docking is analyzed in Campbell (1994), Klose and Drexl (2005), Chen et al. (2006), and Chen (2007). The layout of cross-docking centers is studied in Gue (1999), Bartholdi and Gue (2002), and Lee et al. (2008).

Some other works deal with the vehicle routing as Oh et al. (2006) and Clausen et al. (2009). Regarding operational decisions, some works consider the dock assignment problem: Belle et al. (2012), Boysen and Fliedner (2010), Tsui and Chang (1990), Tsui and Chang (1992), Bozer and Carlo (2008), and Gue (1999).

In a CDC, scheduling decisions are crucial to ensure a rapid turnover and on-time

deliveries. Due to its real-world importance, several truck scheduling works and procedures have been introduced during recent years, treating specific cross-dock settings. For that reason, we highlight in Table 2.1 some outstanding works dealing with truck scheduling problems in a CDC. As proposed by Boysen (2010), a classification scheme for deterministic truck scheduling problems is presented for each work, as well as their contribution.

Table 2.1: Previous specific research works for the truck scheduling problems. Based on Boysen and Fliedner (2010).

Publication	Notation	Complexity	Contribution
Miao et al. (2009)	$[M\|limit, t_{io}\|\star]$	NP-hard	MM, HM, P
Chen and Lee (2009)	$[E2\|t_j = 0\|C_{max}]$	NP-hard	B, ES, P
Chen and Lee (2009)	$[E2\|t_j = 0\|C_{max}]$	NP-hard	P
Chen and Song (2009)	$[E\|t_j = 0\|C_{max}]$	NP-hard	MM, HS, B
Boysen (2010)	$[E\|p_j = p, no - wait, t_j = 0\|\sum T_o]$	Open	MM, HM, ES
Boysen and Fliedner (2010)	$[E\|t_{io}, f_{ix}\|\sum W_s U_s]$	NP-hard	MM, P
Boysen and Fliedner (2010)	$[E\|t_i = 0, f_{ix} \mid \sum W_s U_s]$	NP-hard	P
Boysen et al. (2010)	$[E2\|p_j = p, change\|C_{max}]$	NP-hard	MM, HS, HI, ES, P
McWilliams (2010)	$[E\|no - wait\|*]$	Open	HM
Vahdani and Zandieh (2010)	$[E2\|change\|C_{max}]$	NP-hard	MM, HM
Araújo and Melo (2010)	$[E\|t_j = 0\|C_{max}]$	NP-hard	HM
Arabani et al. (2011)	$[E2\|change\|C_{max}]$	Open	HM
Larbi et al. (2011)	$[E2\|pmtn\|*]$	NP-hard	MM, ES
Alpan et al. (2011)	$[E\|pmtn\|*]$	Open	MM, ES
Lima (2014)	$[E2\|t_j = 0\|\sum C_j^2]$	NP-hard	MM, HS, HM
Cota et al. (2016)	$[E\|t_j = 0\|C_{max}]$	NP-hard	MM, HS

The notation used in column "Contribution" is stated as: MM (mathematical model), HI (heuristic improvement procedure), HM (meta-heuristic), B (bound computation), HS (start heuristic for initial solution), ES (exact solution procedure), P (properties [e.g., complexity] of problem). In the Boysen's tuple notation (Boysen (2010)), column "Notation", the fields are door environment, operational characteristics and the objective, respectively.

Chen and Lee (2009) study a two-machine cross-docking flow shop scheduling problem, in which can process a job at the second machine only after finalizing some jobs at the first one, to minimize the makespan. The authors show the problem is strongly NP-hard. They develop a polynomial approximation algorithm with an error-bound analysis and a branch-and-bound algorithm. Computational results show that the branch-and-bound algorithm can optimally solve problems with up to 60 jobs in a reasonable time. In Chapter 3 we deal with the problem precisely as studied by Chen and Lee (2009) to expand and improve the results obtained by them. For the case of two docks, we develop a hybrid framework that uses the information of Lagrangian multipliers to feed a constructive heuristic. The results show that our method produces deterministic solutions and surpasses the results of previous work.

Chen and Song (2009) extended the problem defined by Chen and Lee (2009) to the 2-stage hybrid cross-docking scheduling problem by considering multiple parallel

processors (multiple docks) per stage (inbound and outbound), allowing simultaneous loading and unloading operations. Based on Johnson's rule, they propose a mixed-integer programming model and four constructive heuristics to investigate the performance for moderate and large-scale instances.

Other studies considering multiple dock environments are proposed by Alpan et al. (2008). Still, with a different aim, a dynamic program-based model is presented to find the schedule of transshipment operations minimizing the total operations cost, solving only small-size instances. Larbi et al. (2009) presented some heuristics to obtain near-optimal solutions for large instances considering the same scenario than Alpan et al. (2008).

Cota et al. (2016) deals with the operational decision problem of scheduling the trucks on multiple inbound and outbound docks. That is, they consider the same setting than Chen and Song (2009). They propose a time-indexed mixed-integer linear programming formulation and a constructive polynomial heuristic. Results are compared to the best heuristics proposed in Chen and Song (2009), Cota et al. (2016) on the medium and large instance, and better solutions are consistently obtained for instances with a greater number of machines and jobs. We also present a generalization of the hybrid approach to the multiple-docks case (Chapter 4), proving to be very efficient when compared against previous works.

The hybrid metaheuristics arise to achieve solutions within a reasonable computational time in very difficult combinatorial optimization problems. The hybrid method consists of a cooperative combination of methods, exact and/or approximated, and aims to absorb to the limit the potentialities of all the approaches (see surveys Alba (2005), Puchinger and Raidl (2005), and Blum et al. (2011)).

Paula et al. (2010) propose a variant of the Lagrangian Relaxation to obtain good bounds to the problem of parallel machines scheduling with sequence-dependent setup times. In this effort, the Lagrangian Relaxation is improved by using a metaheuristic as an internal procedure of the Lagrangian Relaxation. The heuristic is used to generate feasible solutions during the execution of the non-delayed relax-and-cut algorithm.

Pirkwieser et al. (2007) present a similar work, using the Lagrangian Relaxation applied to the Knapsack Constrained Maximum Spanning Tree problem, in which the Lagrangian Multipliers influence the procedures executed by the Genetic Algorithm.

Boschetti and Maniezzo (2009) propose a Lagrangian Metaheuristic procedure. Their approach consists in using metaheuristic techniques to obtain feasible solutions using the information of Lagrangian Multipliers. The solutions obtained tend to be good bounds. We remark that, in these efforts, the potential of the information of the Lagrangian Multipliers is still not completely used. Part of our goal is to improve the use of this information. Readers are referred to Blum et al. (2011) for a survey on

hybrid metaheuristics.

As discussed here, a significant number of CDC articles are dedicated to studying strategic and tactical aspects, not always aligned with industry practice. More recently, Ladier and Alpan (2016a) discuss the relationship between the industry needs and academic research topics, pointing out the gap between both worlds. They highlight some gaps between theory and practice which would need to be filled to get closer to real-life constraints. For example, researchers should remove simplifying assumptions that are difficult to justify in an industrial context. It would be necessary to take deadlines into account for outbound truck departures.

Aiming to develop new perspectives for research concerning cross-docking operations, we create a rescheduling approach that can efficiently solve the cross-docking truck scheduling problem with truck arrival time uncertainty. We expect to reduce the gap between academic research and industrial needs, very well highlighted by Ladier and Alpan (2016a).

2.2 Cross-docking under uncertainty

This section briefly presents the most recent works that address truck scheduling problems in cross-docking centers. We are particularly interested in related papers about cross-docking problems and cross-docking under uncertainty.

As mentioned before, the truck scheduling problem plays an essential role in most cross-docking systems as it affects the efficiency of operations in terms of speed and reliability of deliveries. In static problems, all data are known a priori, and they do not change over time (AL-Behadili (2018)). Some authors define a scheduling problem as static when all machines and jobs are available at time zero (Vieira et al. (2003)), (Gholami et al. (2009)).

In the literature, it is possible to observe several works that generally assume that all problem data (e.g., number of jobs, processing times, release dates, due dates, weights) take constant values. Some of these works are summarized in Table 2.1.

More recently, in Gaudioso et al. (2020) the authors propose a Lagrangian heuristic algorithm for a cross-docking problem. They consider a multi-door truck scheduling and transshipment with constant processing time and develop Lagrangian decomposition in three sub-problems for the integer linear model. The algorithm inserted two heuristics to solve the Lagrangian Dual and calculate the lower and upper bounds. The results found were entirely satisfactory.

A novel approach for solving the multi-dock truck-sequencing problem, considering a parallel machine instead of a flow-shop environment, is presented in Nogueira et al.

(2020). The authors propose a polynomial constructive heuristic that can generate primal and dual bounds obtaining good gaps for the makespan objective, outperforming the time-indexed mathematical formulation and state-of-the-art polynomial heuristics. Both articles consider static environments.

In dynamic problems, not all the data is known in advance, planning is done, and once operations are started, changes in the parameters are considered; for example, a new job may arrive and should be included in the execution, or a machine breaks down during the operation. In these two cases, the scheduling needs to be redone (Ouelhadj and Petrovic (2009)). Some authors define dynamic scheduling for the case where jobs are not immediately available at time zero. They can also classify dynamic problems in deterministic cases (arrival times are known a priori in the future) and stochastic (not known, may or may not have a known probability distribution a priori) (Nie et al. (2013) and Potts and Strusevich (2009)).

Scheduling under uncertain conditions, termed dynamic scheduling, is of great importance for successfully implementing real-world cross-docking systems. Uncertainties can be related to resources, such as machine breakdown, delay in the arrival or shortage of materials, and job-related parameters, such as job cancellation, early or late arrival of jobs, and changes in job processing time. Dynamic scheduling has been defined under three categories: reactive scheduling, predictive-reactive scheduling, and robust pro-active scheduling. The most well-known efficient approach used in dynamic scheduling is the predictive-reactive approach Ouelhadj and Petrovic (2009).

In practice, scheduling decisions at a CDC are subject to several real-time events, hence the growing need to study alternatives to deal with uncertainties methodologies. Developing approaches for cross-docking problems have been given considerable effort in the literature in recent years. However, only a few works successfully deal with real-life cross-docking problems, while most articles rely on theoretical and simplifying assumptions. Therefore, implementing such approaches is usually impractical for real-world systems, given the increased complexity we are immersed in. This particular situation is already pointed out in Ladier and Alpan (2016a). Their work still highlights some gaps between the literature and industry practices, which would need to be filled by future research.

Most real cross-docking systems are subjected to perturbations from different sources of uncertainties. According to Boysen and Fliedner (2010), truck arrival times are very often subject to unexpected events caused by various reasons, such as accidents, traffic jams, truck problems, among others. These unexpected events can cause delays in the arrival of trucks to the cross-docking centers, and consequently, delays in customer deliveries. Considering these unexpected situations and possible disruptions motivated us to develop a rescheduling approach for solving multi-dock truck scheduling problems

where truck arrival time is uncertain.

In Ladier and Alpan (2016b), the authors propose robust methods for cross-dock scheduling with time windows. They developed two robust project scheduling methods and concluded that the resource redundancy methodology gives good results in the cross-docking problem. The truck scheduling at a cross-dock center in case of uncertain truck arrival times is studied in Konur and Golias (2013). In their paper, each truck's arrival time is unknown, and time windows are considered. They analyze and compare three approaches: deterministic, pessimistic, and optimistic, and a Genetic Algorithm is proposed for the single and bi-level problems.

The truck scheduling at a cross-dock facility in case of uncertain truck arrival times is studied by Konur and Golias (2013). Their study considers that the cross-dock operator only acknowledges the arrival time window of each truck. They analyze and compare three different approaches: deterministic (which assumes expected truck arrival times are equal to their mid-arrival time windows), pessimistic (which assumes the worst truck arrivals will be realized), and optimistic approach (which assumes the best truck arrivals will be realized). A single and bi-level optimization problem is formulated, and a Genetic Algorithm and its modification are discussed for the single and bi-level optimization problems. Numerical studies show that a hybrid approach regarding the pessimistic and the optimistic approaches may outperform all three approaches in some instances.

Our purpose with this work is to study cross-docking problems. The specific goals involve developing models, algorithms, and resolution methods for the truck scheduling problem in cross-docking centers, adding aspects of uncertainty and developing methodologies that efficiently solve the problem in a more realistic environment. We started our studies considering the 2-dock cross-docking flow shop scheduling problem (Chapter 3). In Chapter 4 we developed a generalization of the hybrid approach to the multiple-docks case, proving to be very efficient when compared against previous work. Finally, in Chapter 5, we reshaped how we look to the future of logistics strategies, especially to cross-docking problems. Therefore, we propose a rescheduling approach for the multi-dock truck scheduling problem under truck arrival time uncertainty. We hope to help fill a critical gap between the current state-of-the-art and the observed industry practice: scheduling operations under uncertain or late truck arrivals.

Chapter 3

The 2-dock cross-docking flow shop scheduling problem

3.1 Introduction

Considering a 2-dock cross-docking center, in which n loaded trucks arrive with products demanded by one or more customers. In the center, each truck must unload its cargo and load it into one or more outbound trucks, responsible for delivery to specific destinations in the supply chain. Each departing vehicle may leave the center only after the cargo is fully loaded, and it can begin the loading process only after all needed products were available in the center. Here we consider the existence of two docks in the center, one dock dedicated to unload the trucks and the other one for loading purposes, machine 1 (M1) and machine 2 (M2), respectively. The problem corresponds to define the truck sequence in the inbound and outbound dock, minimizing the completion time of the last job processed by machine 2.

3.2 Mathematical Model

In this section, we present a mathematical formulation for a 2-dock version based on scheduling problem initially proposed by Chen and Lee (2009). The integer programming model considered in this work adopts a time-indexed formulation proposed by Lima (2014) but adopting the makespan as objective function, as proposed by Chen and Lee (2009).

The following parameters, sets and variables are considered:

Input Parameters

- n: number of jobs to be processed on M1.
- m: number of jobs to be processed on M2.
- p_{ij}: processing time of job j on machine i.
- T_f: size of time horizon. A first estimate for the time horizon is the sum of the processing times of all jobs.
- $T_0 \geq 0$: beginning of time horizon for the jobs $j \in J^2$, i.e., minimum starting date for processing of the output trucks, initially set as $T_0 = min_{j \in J^2} \{\sum_{i \in J^1, i \in S_j} p_{1i}\}$, then $T = \{T_0, ..., T_f\}$ $\forall j \in J^2$.

Sets

The set of periods is defined as $T = \{0, ..., T_f\}$. To represent arrivals and truck departures in cross-docking center, two sets of jobs are created:

- $J^1 = \{j_1^1, j_2^1, ..., j_n^1\}$, set of n jobs (inbound trucks), which must be processed on M1.
- $J^2 = \{j_1^2, j_2^2, ..., j_m^2\}$, set of m jobs (outbound trucks), which must be processed on M2.
- S_j, set of precedent jobs. For each job $j_j^2 \in J^2$, there is a corresponding subset of J^1 that must be completed before beginning its process. It is considered that each element job $j \in J^2$ has at least one job S_j as precedent.

Decision variables

- x_{jt} $(\forall j \in J^1, \forall t \in T)$, binary variable, x_{jt} is equal to 1 if inbound job j starts its process at time t and equal to 0 otherwise.
- y_{jt} $(\forall j \in J^2, \forall t \in T)$, binary variable, y_{jt} is equal to 1 if outbound job j starts its process at time t and equal to 0 otherwise.
- C_{max}: makespan. Maximum completion time in M2.

The complete mathematical model (F) is presented as follows:

$$(F) \qquad Minimize\ C_{max} \tag{3.1}$$

subject to

$$\sum_{t=0}^{T_f-p_{1j}} x_{jt} = 1, \qquad \forall j \in J^1, \tag{3.2}$$

$$\sum_{t=T_0}^{T_f-p_{2j}} y_{jt} = 1, \qquad \forall j \in J^2, \tag{3.3}$$

$$\sum_{j\in J^1} \sum_{s=max(0;t-p_{1j}+1)}^{t} x_{js} \leq 1, \qquad \forall t \in T, \tag{3.4}$$

$$\sum_{j\in J^2} \sum_{s=max(T_0;t-p_{2j}+1)}^{t} y_{js} \leq 1, \qquad \forall t \in T, \tag{3.5}$$

$$\sum_{t=T_0}^{T_f-p_{2j}} t y_{jt} - \sum_{t=0}^{T_f-p_{1k}} (t+p_{1k}) x_{kt} \geq 0, \qquad \forall j \in J^2, \forall k \in S_j, \tag{3.6}$$

$$C_{max} \geq p_{2j} + \sum_{t=T_0}^{T_f-p_{2j}} t y_{jt}, \qquad \forall j \in J^2, \tag{3.7}$$

$$x_{jt} \in \{0,1\}, \qquad \forall j \in J^1, \forall t \in T, \tag{3.8}$$

$$y_{jt} \in \{0,1\}, \qquad \forall j \in J^2, \forall t \in T, \tag{3.9}$$

$$C_{max} \geq 0. \tag{3.10}$$

The objective function (3.1) minimizes the makespan. The set of constraints (3.2) ensures that each job $j_j^1 \in J^1$ should start its processing in one and only one period within the time horizon. The set (3.3) works with the same reasoning on M2. Constraints (3.4) ensure that a job $j_j^1 \in J^1$ does not start its processing while another job is being processed in the same machine M1. The constraints set (3.5) works, in the same way, but applied to jobs $j_j^2 \in J^2$. The set (3.6) controls the precedence relationships. For every existing precedence relation, the start date of the job $j_j^2 \in J^2$ should be greater than or equal to the completion time of its precedent $j_k^1 \in S_j$. The set of constraints (3.7) indicates that the variable C_{max} should be the maximum completion time of jobs in J^2. Finally, sets (3.8) to (3.10) define the variables' domain.

3.3 Lagrangian Relaxation

Let L(λ) be the relaxation of formulation F, when constraints 3.6 are dualized and its violation penalized in the objective function. For each constraint is associated one Lagrangian multiplier λ representing the weight given to the violation. The set of Lagrangian multipliers will be represented by λ_{jk}, with $j \in J^2$ and $k \in S_j$. Notice that, these constraints are the cross-docking precedence relations, coupling decisions in M1 with decisions in M2. Thus, the subproblem L(λ) is defined as:

$$L(\lambda) = Minimize\ C_{max} + \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \Big(\sum_{t=0}^{T_f - p_{1k}} (t + p_{1k}) x_{kt} - \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt} \Big) \tag{3.11}$$

s.t. (3.2) – (3.5), (3.7) – (3.10), $\lambda_{jk} \geq 0$.

Now we are able to uncouple the problem into two new scheduling problems, one for each machine. Subproblem X considers M1 while subproblem Y considers M2. The value of the lower bound is obtained by adding the subproblems objective functions L$(\lambda)_x$ and L$(\lambda)_y$.

Subproblem X

Isolating the terms of the objective function that contain variables x, we reach to:

$$L(\lambda)_x = Minimize \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=0}^{T_f - p_{1k}} (t + p_{1k}) x_{kt} \tag{3.12}$$

s.t. (3.2), (3.4), (3.8), and $\lambda_{jk} \geq 0$.

Rewriting the above objective function from the perspective of jobs in J^1, we have:

$$L(\lambda)_x = Minimize \sum_{j \in J^1} \sum_{t=0}^{T_f - p_{1j}} (t + p_{1j}) x_{jt} w_j^1 \tag{3.13}$$

where $w_j^1 = \sum_{i \in J^2} \lambda_{ij}$, where $\lambda_{ij} = 0$ if $j \notin S_i$, .

The problem mentioned above is known as The Total Weighted Completion Time (Pinedo (2008)), denoted by $1||\sum C_j W_j$. This problem can be solved by the WSPT rule (Weighted Shortest Processing Time First) proposed by Smith (1956). According to this rule, the optimal solution is obtained by ordering the jobs in descending order of w_j^1 / p_{1j}, and can be obtained in O($n \log(n)$).

Subproblem Y

With the terms associated to variables y, subproblem Y can be defined as:

$$L(\lambda)_y = Minimize\ C_{max} - \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt} \tag{3.14}$$

subject to

$$\sum_{t=T_0}^{T_f - p_{2j}} y_{jt} = 1, \qquad \forall j \in J^2, \tag{3.15}$$

$$\sum_{j \in J^2} \sum_{s=max(T_0; t-p_{2j}+1)}^{t} y_{js} \leq 1, \qquad \forall t \in T, \tag{3.16}$$

$$C_{max} \geq p_{2j} + \sum_{t=T_0}^{T_f - p_{2j}} t y_{jt}, \qquad \forall j \in J^2, \tag{3.17}$$

$$y_{jt} \in 0, 1, \qquad \forall j \in J^2, \forall t \in T, \tag{3.18}$$

$$\lambda_{jk} \geq 0, \qquad \forall j \in J^2, k \in S_j, \tag{3.19}$$

$$C_{max} \geq 0. \tag{3.20}$$

Defining $w_j^2 = -\sum_{k \in S_j} \lambda_{jk}\ \forall j \in J^2$, we can rewrite the problem as follows:

$$L(\lambda)_y = Minimize\ C_{max} + \sum_{j \in J^2} \sum_{t=T_0}^{T_f - p_{2j}} t.y_{jt} w_j^2 \tag{3.21}$$

s.t. (3.15), (3.16), (3.17), (3.18), and (3.20).

The first term of the objective function (3.21) is the makespan. The second term represents the sum of the weighted starting times of the jobs on M2, also known as The Total Weighted Starting Time, denoted $\sum I_j W_j$. Thus, the Y subproblem can be defined as $1||C_{max} + \sum I_j W_j$. To solve this subproblem, we proposed a rule named WSPT-TRD.

The WSPT-TRD rule works as follows. First, the WSPT rule is used to generate a sequence of jobs on M2. The second step is to define the correct allocation of jobs in the time horizon. For that, we evaluate the increase of the makespan when the weights associated with the jobs are negative. When the weights are negative, it creates a trade-off situation, as explained next.

Generally, weights w_j^2 may have positive or negative values. Let us first consider schedule decisions for jobs with $w_j^2 \geq 0$, both C_{max} and $\sum I_j W_j$ have similar behavior since the objective function is minimized by allocating the jobs at the beginning of the

time horizon. In our particular case, as $\lambda_{jk} \geq 0$, we do not have the case of $w_j^2 > 0$; however, our algorithm here proposed considers this possibility.

Considering jobs with negative weights ($w_j^2 < 0$), the objective function terms lead to different solutions. On the one hand, by considering the makespan C_{max} one tends to allocate the jobs at the beginning of time horizon, as the weights do not influence its final value. On the other hand, when analyzing the $\sum I_j W_j$ one tends to allocate the jobs as late as possible, taking advantage of the negative weights.

We notice that if we delay all jobs with negative weights one-time unit, the C_{max} increases one unit. Thus, our algorithm computes the sum of the negative weights, if the sum of the negative weights is less than -1, the displacement of the jobs towards the end of the time horizon compensates the increase of C_{max}, defining in this way the best allocation of jobs.

Pseudocode of algorithm WSPT-TRD:

Step 1: Compute the weights w_j^2 on the machine 2 ($w_j^2 = -\sum_{k \in S_j} \lambda_{jk} \; \forall j \in J^2$).

Step 2: Sort the jobs in J^2 in decreasing order of w_j^2 / p_{2j}.

Step 3: Compute the sum of the negative weights as $\delta = \sum_{j \in J^2} w_j^2 \; \forall j \in J^2 \wedge w_j^2 < 0$.

Step 4: If $w_j^2 = 0 \; \forall j \in J^2$ or $\delta \geq -1$, allocate the jobs from $t = T_0$ by sequence generated by WSPT rule.

Step 5: If there exists $w_j^2 < 0$ and $\delta \leq -1$, allocate the jobs from t $= T_f$ by the reverse sequence generated by WSPT rule.

Considering the problem $1||C_{max} + \sum I_j W_j$, the algorithm WSPT-TRD obtains an optimal solution. The proof is provided in Appendix B.

3.3.1 Lower Bounds

Besides the lower bound obtained from the Lagrangian relaxation, we consider here two other lower bounds. The first one, called LB_1 is defined as proposed by Chen and Lee (2009), as follows:

$$LB_1 = \sum_{i \in J^2} p_{2i} + min_{j \in J^2} \{ \sum_{i \in J^1, i \in S_j} p_{1i} \}. \tag{3.22}$$

LB_1 basically computes the sum of processing time on $M2$, plus the minimum amount of time necessary to begin the process of the first job in $M2$, that is, the minimum set of precedent jobs in $M1$.

The second lower bound, LB_2, proposed in this work, considers a complementary condition:

$$LB_2 = \sum_{i \in J^1} p_{1i} + min_{j \in J^1}\{ \sum_{i \in J^2, i \in P_j} p_{2i}\}. \tag{3.23}$$

Where, P_j represents the successors subsets jobs $j \in J^1$ corresponding to job $i \in J^2$, i.e., successors are all jobs that must be processed after completion for a given job in the first stage.

In this case, LB_2 computes the sum of processing times in $M1$ plus the minimum set of common successors of $j \in J^1$. Finally, the Lower Bound is defined as the maximum lower bound, $LB = max\{LB_1, LB_2\}$.

It is worth noticing that with a valid LB we can redefine a new date for the beginning of the processing time of jobs in the $WSPT-TRD$ algorithm. Particularly, in Step 4, $T_0 = LB - \sum_{i \in J^2} p_{2i}$, this is valid for the jobs with $w_j^2 = 0$ or for all jobs when $\delta \geq -1$.

3.4 Hybrid Lagrangian Metaheuristic

In the Hybrid Lagrangian Metaheuristic Framework proposed in this work, the Lagrangian Dual is solved by the Volume algorithm, as proposed in Barahona and Anbil (2000) and Nogueira (2014). Readers are referred to Barahona and Ladányi (2006) and Fukuda (2007) for a discussion on the Volume algorithm and its performance. In our case, the Lagrangian Relaxation incorporates heuristics as internal procedures with a focus on obtaining feasible solutions. These heuristics are based on a constructive method and a Local Search. Both procedures are executed sequentially under the Lagrangian Relaxation.

The Volume algorithm is an extension of the subgradient algorithm, which produces a sequence of primal and dual solutions, thus being able to prove optimality. This algorithm has similar computational effort than the subgradient algorithm, and it presents similarities with the Conjugate Subgradient method Lemaréchal (1975), Wolfe (1975) and the Bundle method Lemaréchal (2001, 572). A discussion of its main features and a global convergence analysis can be found in Bahiense et al. (2002).

In the proposed algorithm the precedence constraints are dualized. The Lagrangian multipliers define a penalty to a given job allocated at a given position. If the job has a large associated value, it means that it has a greater impact on the objective function, i.e., it is allocated in a disadvantageous position. This information can be used to decide when to schedule the jobs. Following, we describe two constructive heuristics used to obtain feasible solutions.

3.4.1 Constructive Heuristics

The heuristic H1 uses information from Lagrangian multipliers to sort jobs on M1, while H2 uses information from Lagrangian multipliers to sort jobs on M2. In the final step both heuristics use an NEH-like heuristic (Nawaz et al. (1983)) to construct the feasible solution. For a given list of jobs, the NEH function constructs a solution by scheduling the jobs on the list one by one, in the best position of the partial schedule. The two heuristics are described below:

H1

Step 1: For each job on the machine 1 ($k \in J^1$), compute $\beta_k = \frac{\sum_{j \in J^2} \lambda_{jk}}{Nprec_k}$.

In which $Nprec_k$ corresponds to the number of jobs in the second stage that are waiting for the completion of job k on M1.

Step 2: Get the sequence on M1 by ordering jobs in decreasing order of β_k.

Step 3: Calculate the new release dates (r_j, $j \in J^2$) of jobs on M2.

Step 4: Get the sequence on M2 by ordering jobs in non-decreasing order of release dates r_j and apply NEH.

H2

Step 1: For each job on M2 ($j \in J^2$), compute $\beta_j = \frac{\sum_{k \in S_j} \lambda_{jk}}{Nprec_j}$.

Where $Nprec_j$ corresponds to the number of precedents that job j has.

Step 2: Get the sequence on M2 by ordering jobs in increasing order of β_j.

Step 3: Sequence the jobs on the M1 according the sequence on M2, respecting the precedence relations of cross-docking.

If there is a job on M1 without precedence relationship in the machine 2, schedule this job last.

Step 4: Calculate the new release dates (r_j, $j \in J^2$) of jobs on M2, from the sequence on M1.

Step 5: Get the sequence on M2 by ordering jobs in non-decreasing order of release dates r_j and apply NEH.

3.4.2 The Lagrangian algorithm

Let x and y be solutions of subproblems X and Y, respectively, with objective function value z. Then λ are the Lagrangian multipliers obtained by the Volume Algorithm, and $\nu(\lambda,\ x,\ y)$ is the subgradient. Let UB be the upper bound or feasible solution. The initial UB is generated by a Lagrangian metaheuristic. The algorithm is described below:

Step 0: We start with a null vector λ_{jk} (λ_{jk}=0). Solve the Lagrangian subproblem to obtain the initial solution of X subproblem (x^0), Y subproblem (y^0) and objective function value (z^0). Let UB be the feasible solution obtained by Lagrangian metaheuristic. Set $\bar{x} = x^0$, $\bar{y}$=y^0, $\bar{z} = z^0$ and k=1.

Step 1: Compute the subgradient $\nu(\lambda_{jk-1},\ \bar{x},\ \bar{y})$ and λ_{jk}=$\bar{\lambda}_{jk}$ + $s\nu(\lambda_{jk-1})$, the calculation of the step size s is given by the equation (3.27). Solve the Lagrangian subproblem with the new λ_{jk} and let x^k, y^k and z^k be the solutions obtained. Then $\bar{x}$ = αx^k + (1-α)$\bar{x}$, $\bar{y}$ = αy^k + (1-α)$\bar{y}$ and $\bar{\lambda}$=$\alpha\lambda^k$ + (1-α)$\bar{\lambda}$, where α is a number between 0 and 1, defined by convex combination such as defined in (3.24).

Step 2: If $z^k > \bar{z}$ update $\bar{\lambda}$=λ_{jk} and $\bar{z}$=z^k. If $\bar{z}$ improves by 10% since the last run of the Lagrangian metaheuristic then go to Step 3. Else go to Step 4.

Step 3: Lagrangian heuristics (H1, H2):

1. $UB^k \leftarrow$ Constructive Heuristics (H1 and H2);
2. $UB^k \leftarrow$ Local Search (list);

If $UB^k < UB$ update UB = UB^k and add a cut in order to reduce the time horizon, improving the limits found (update T_f=UB).

Step 4: Stop criteria: If satisfied stop. Else let $k = k + 1$ and go to Step 1.

The step size s and the parameter α used to define $\bar{x}$ and $\bar{y}$, are computed as proposed in Barahona and Anbil (2000) and Fukuda (2007). First, we define α_{opt} as:

$$\alpha_{opt} = argmin \parallel \alpha\nu'^{k}_{(\lambda_{jk-1},x^k,y^k)} + (1-\alpha)\nu^{k}_{(\lambda_{jk-1},\bar{x},\bar{y})} \parallel^2 \tag{3.24}$$

The parameter values π, MaxWaste, factor, α_{max}, st, α, yellow and green, some of them yet to be introduced, were defined using the SPOT method, as explained in Appendix C. The parameters α_{max} and α are initially defined as 0.3034 and 0.0830

respectively, based on computational tests performed by SPOT. And α is computed as:

$$\alpha = \alpha_{max} * \alpha \quad if \quad \alpha_{opt} < 0 \tag{3.25}$$

$$\alpha = min\{\alpha_{opt}, \alpha_{max}\} \quad if \quad \alpha_{opt} \geq 0 \tag{3.26}$$

Before setting the value of the step s, we need to define the parameter π. Therefore, to set the value of π we define three types of algorithms iterations as defined in Barahona and Anbil (2000) and Barahona and Ladányi (2006).

Each iteration with no improvement is named red. Then, the parameter MaxWaste represents the maximum number of iterations without a lower bound improvement. If $z^k > \bar{z}$ and $\nu'^k_{(\lambda_{jk-1},x^k,y^k)}\nu^k_{(\lambda_{jk-1},\bar{x},\bar{y})} < 0$, it means that a longer step in the direction to ν^k would have given a smaller value for z^k. Those iterations are denominated *yellow*, otherwise, the iteration is denominated *green*. At each *yellow* iteration we would multiply π by the yellow parameter. At each *green* iteration we would multiply π by the green parameter. After a sequence of 24 consecutive *red* iterations, we would multiply π by factor parameter. Thus, the step size s at iteration k is defined as:

$$s = \frac{\pi * (st * UB - \bar{z}))}{\parallel \nu_{(\lambda_{jk-1},\bar{x},\bar{y})} \parallel^2} \tag{3.27}$$

In the "Lagrangian heuristics step" two constructive heuristics and a Local Search are executed sequentially. The Local Search is implemented based on the proposals of Arroyo et al. (2009) and Stutzle (1998). The procedure adopted is composed by "swap" and "insertion" moves in the sequence on machine 2 generated by H1 and H2. The former consists of interchanging all pairs of jobs. The latter consists of removing a job from its original position and inserting it on one of the $n-1$ remaining positions. The local search procedure stops when it is unable to improve the solution further.

Finally, the stop criterion is determined if one of the following criteria is satisfied:

1. Maximum number of iterations, in this case 1000 iterations, or;
2. Relative tolerance GAP defined as: $\frac{z^k - \bar{z}}{\bar{z}} < 0.1$, or;
3. Null Subgradient module or negligible: $\parallel \nu_{(\lambda_{jk-1},\bar{x},\bar{y})} \parallel^2 \leq 0.000001$.

3.4.3 JB Heuristic

We compare the results of heuristics H1 and H2 with the best heuristic introduced by Chen and Lee (2009), which is based on Johnson's algorithm. The heuristic JB is

proposed in two steps. Firstly, for an instance of $F2|CD|C_{max}$, the authors construct an instance of $F2||C_{max}$ with n jobs. The first stage is maintained unchangeable and the second stage is converted into n jobs. Secondly, Johnson's algorithm is applied to obtain a sequence in the first stage, generating a lower bound. From the sequence in the first stage, jobs $j \in J_2$ at the second stage are sequenced as soon as possible in the time horizon, respecting the completion time of its predecessors. We run the above algorithm for the primary problem and its reverse problem and select the best solution. As a result, an Upper Bound for the problem is generated (See Chen and Lee (2009) for a more detailed explanation), so we calculated the relative GAP. With the objective of a fair comparison, the NEH algorithm and Local search are used to refine the results of JB in the same way than used in H1 and H2.

3.4.4 CDH Heuristic

The proposed H1 and H2 algorithms are also compared with the heuristic in Cota et al. (2016), called CDH. The CDH heuristic starts by creating a preliminary schedule in the second machine. With this, a schedule is defined in the first machine and finally the second machine is rescheduled. The heuristic initially creates fictitious processing times $TF_j = (\sum_{i \in J^1, i \in S_j} p_{1i}) + p_{2j}$ for each job $j \in J^2$. Then, a preliminary schedule on M2 is obtained by ordering jobs in increasing order of fictitious processing times. Then the jobs on the M1 are sequence according the preliminary schedule on M2, respecting the precedence relations of cross-docking. The sequence on M1 imposes release dates to the jobs in the second machine due to precedence relationships. So, the ERD rule (earliest release date first) is applied to reschedule jobs in the second machine, obtaining a sequence on the M2. Let UB be the upper bound obtained with the heuristic and LB be the lower bound calculated as proposed in Chen and Song (2009), we generate the percentage GAP for all tested instances.

3.5 Results for the 2-dock problem

To investigate the performance of the Complete Model and Hybrid Lagrangian Metaheuristic Framework, artificial instances are generated varying the processing time of jobs and the number of jobs on machines 1 and 2, according to Chen and Lee (2009). We ran the tests on a single thread in the Intel Xeon-Silver 4110 (2.1GHz/8-core/85W) with 64 GB memory and Linux operational system. The programming language used is C++ with the optimization software CPLEX 12.4.

Table 3.1: Summary of the artificial benchmark.

Group	Jobs M1(n)	Jobs M2(m)	NP	TP
1	5	3-4-5-6-7	U(1,4)	U(1,10)
	10	6-8-10-12-14	U(1,9)	U(1,10)
	20	12-16-20-24-28	U(1,19)	U(1,10)
	40	24-32-40-48-56	U(1,39)	U(1,10)
	60	36-48-60-72-84	U(1,59)	U(1,10)
2	5	3-4-5-6-7	U(1,4)	U(10,100)
	10	6-8-10-12-14	U(1,9)	U(10,100)
	20	12-16-20-24-28	U(1,19)	U(10,100)
	40	24-32-40-48-56	U(1,39)	U(10,100)
	60	36-48-60-72-84	U(1,59)	U(10,100)

3.5.1 Instances Generation

The instances used in this work are generated through the software MATLAB, following the description found in Chen and Lee (2009). Table 3.1 presents a summary of the generated instances and its characteristics. It is divided into two groups, each row informs the features of a sub-group of instances. n and m indicate the number of jobs in each stage. (NP) the discrete uniform distribution to select the number predecessors of jobs $j \in J^2$ and (TP) the distribution to generate all processing times.

In each group of instances, n is fixed and the number m is randomly chosen from the range of $[0.6n, 0.8n, 1.0n, 1.2n, 1.4n]$. For each n, five instances with different m values are considered. And for each pair (n, m), we generate 10 different problems, therefore the benchmark consists of 500 instances, 50 instances per row of Table 3.1 [1].

3.5.2 Computational Results

In Table 3.2 we report computational results for the Complete Model (MIP) and heuristics JB, CDH, H1, and H2. Columns n and m show respectively the number of jobs in the first and second machine. For each instance size (n, m) we report the best result and the average of 10 cases tested. We also presented the subgroup's average results.

The GAP(%) is computed as $\frac{\text{(Upper Bound - Lower Bound)}}{\text{Upper Bound}}$ and the column T(s) refers to CPU time expended to solve the problem in seconds. The run time is limited to one hour of CPU time (3.600 seconds), and results are depicted in Table 3.2 are registered. The dash (-) means that the corresponding value is not found. Finally, after

[1] all instances and codes are available at https://github.com/GabrielaBragaFonseca/Cross-docking-Problems

comparing and analyzing the MIP, JB, CDH, H1 and H2 the best results obtained are highlighted in the Table.

The GAP is calculated considering the best lower bound found when comparing the lower bounds of Complete Model and Hybrid Lagrangian Metaheuristic Framework. It is worth highlighting that the constructive heuristic JB is proposed by Chen and Lee (2009) and CDH heuristic is proposed by Cota et al. (2016).

The MIP results show that the proposed model is efficient to solve to optimality only small instances. Furthermore, for the larger instances the model found no solution. These results expose the difficulty of solving time-indexed models. In the time-indexed problems, the number of variables is proportional to the time horizon, and the higher the number of jobs the greater the horizon of time to sequence them, increasing the computational effort to solve these problems. This fact justifies the proposal of the hybrid constructive heuristics to solve instances with a greater number of jobs.

Results show that H1 and H2 performs better than heuristics JB and CDH. Based on the Table 3.2, the heuristic H2 showed GAP results more efficients. Comparing the average GAPs we can observe that the heuristic H2 has its better performance when $n < m$ for small and large instances. When $n > m$, the heuristic H2 gets better for small cases, while H1 presents better results for large instances. The same behavior is noticed when $n = m$, H2 is best for small cases, and H1 for larger ones. The heuristic JB had no best GAP in any instance. As for the CDH heuristic, the GAPs found were high, which was already expected, since the heuristic was developed for a more general machine's scenario.

Table 3.2: Computational results for complete model and constructive heuristics.

			Group 1 - Processing time [1, 10]										Group 2 - Processing time [10, 100]									
n	m		MIP		JB		CDH		H1		H2		MIP		JB		CDH		H1		H2	
			GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)	GAP	T(s)
5	3	Best	0.0%	0.1	**0.0%**	0.0	15.2%	0.0	**0.0%**	0.0	**0.0%**	0.0	0.0%	6.7	**0.0%**	0.0	7.8%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	0.2	6.8%	0.0	26.3%	0.0	**4.8%**	0.0	**4.8%**	0.0	0.0%	21.0	2.4%	0.0	28.8%	0.0	**2.3%**	0.0	**2.3%**	0.0
	4	Best	0.0%	0.1	**0.0%**	0.0	2.7%	0.0	**0.0%**	0.0	**0.0%**	0.0	0.0%	3.9	**0.0%**	0.0	3.0%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	0.3	7.7%	0.0	22.3%	0.0	**5.9%**	0.0	**5.9%**	0.0	0.0%	354.9	7.5%	0.0	24.2%	0.0	**6.2%**	0.0	**6.2%**	0.0
	5	Best	0.0%	0.3	4.5%	0.0	17.1%	0.0	**0.0%**	0.0	**0.0%**	0.0	0.0%	32.6	3.8%	0.0	19.3%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	0.5	9.8%	0.0	32.8%	0.0	**2.5%**	0.0	3.3%	0.0	0.8%	1570.1	9.9%	0.0	33.4%	0.0	**2.6%**	0.0	**2.6%**	0.0
	6	Best	0.0%	0.1	4.8%	0.0	13.0%	0.0	**0.0%**	0.0	**0.0%**	0.0	0.0%	80.9	5.7%	0.0	13.5%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	1.3	16.6%	0.0	29.4%	0.0	**5.2%**	0.0	5.5%	0.0	11.8%	2388.3	16.1%	0.0	30.7%	0.0	**4.8%**	0.0	6.2%	0.0
	7	Best	0.0%	0.2	**0.0%**	0.0	16.7%	0.0	**0.0%**	0.0	**0.0%**	0.0	0.0%	94.9	**0.0%**	0.0	16.6%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	3.7	14.1%	0.0	36.4%	0.0	6.3%	0.0	**5.4%**	0.0	13.1%	2882.5	13.9%	0.0	38.0%	0.0	**5.1%**	0.0	6.7%	0.0
Subgroup Average			0.0%	1.2	11.0%	0.0	29.4%	0.0	**4.9%**	0.0	5.0%	0.0	5.1%	1443.4	10.0%	0.0	31.0%	0.0	**4.2%**	0.0	4.8%	0.0
10	6	Best	0.0%	5.1	**0.0%**	0.0	7.5%	0.0	**0.0%**	0.0	**0.0%**	0.0	28.7%	2801.1	0.7%	0.0	8.9%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	20.2	11.8%	0.0	24.8%	0.0	1.8%	0.0	**1.7%**	0.0	28.2%	3509.9	11.3%	0.0	23.0%	0.0	**1.4%**	0.0	**1.4%**	0.0
	8	Best	0.0%	10.2	2.7%	0.0	9.6%	0.0	**0.0%**	0.0	**0.0%**	0.0	23.6%	3597.6	2.3%	0.0	15.0%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	0.0%	167.5	13.3%	0.0	25.8%	0.0	6.0%	0.0	**5.4%**	0.0	36.7%	3600.0	17.0%	0.0	30.5%	0.0	**3.7%**	0.0	4.2%	0.0
	10	Best	0.0%	14.1	1.4%	0.0	14.8%	0.0	**0.0%**	0.0	2.2%	0.0	35.7%	3600.0	2.2%	0.0	26.4%	0.0	4.6%	0.0	**1.0%**	0.0
		Average	0.0%	369.6	15.9%	0.0	28.8%	0.0	6.5%	0.0	**6.4%**	0.0	45.2%	3600.0	18.4%	0.0	36.8%	0.0	**7.6%**	0.0	7.9%	0.0
	12	Best	0.0%	42.1	3.9%	0.0	17.6%	0.0	**0.0%**	0.0	**0.0%**	0.0	51.7%	3600.0	3.1%	0.0	19.3%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	5.0%	2440.5	14.0%	0.0	41.0%	0.0	5.4%	0.0	**5.3%**	0.0	65.2%	3600.0	13.9%	0.0	40.8%	0.0	**5.4%**	0.0	**5.4%**	0.0
	14	Best	0.0%	21.9	1.3%	0.0	27.9%	0.0	**0.0%**	0.0	1.0%	0.0	53.2%	3600.0	2.0%	0.0	28.9%	0.0	**0.0%**	0.0	**0.0%**	0.0
		Average	4.6%	3141.6	10.8%	0.0	45.5%	0.0	**7.0%**	0.0	8.1%	0.0	73.6%	3600.0	11.4%	0.0	45.2%	0.0	6.7%	0.0	**5.8%**	0.0
Subgroup Average			1.9%	1227.9	13.1%	0.0	33.2%	0.0	**5.3%**	0.0	5.4%	0.0	49.8%	3582.0	14.4%	0.0	35.2%	0.0	5.0%	0.0	**4.9%**	0.0
20	12	Best	1.5%	3600.0	10.0%	0.0	12.4%	0.0	1.5%	0.0	**0.7%**	0.0	-	3600.0	10.2%	0.0	13.7%	0.0	**0.6%**	0.0	**0.6%**	0.0
		Average	14.8%	3600.0	18.1%	0.0	26.1%	0.0	**6.6%**	0.0	7.1%	0.0	-	3600.0	17.9%	0.0	26.0%	0.0	**6.4%**	0.0	6.5%	0.0
	16	Best	14.5%	3600.0	16.4%	0.0	21.4%	0.0	**1.6%**	0.0	5.4%	0.0	-	3600.0	17.4%	0.0	22.7%	0.0	**2.3%**	0.0	4.2%	0.0
		Average	32.4%	3600.0	23.0%	0.0	32.6%	0.0	**9.8%**	0.0	10.5%	0.0	-	3600.0	23.3%	0.0	32.6%	0.0	**8.2%**	0.0	10.4%	0.0
	20	Best	24.5%	3600.0	14.3%	0.0	31.9%	0.0	**2.5%**	0.0	4.4%	0.0	-	3600.0	15.5%	0.0	31.2%	0.0	3.7%	0.0	**1.3%**	0.0
		Average	36.0%	3600.0	24.2%	0.0	38.5%	0.0	11.1%	0.0	**10.9%**	0.0	-	3600.0	24.8%	0.0	38.0%	0.0	11.4%	0.0	**10.3%**	0.0
	24	Best	19.9%	3600.0	2.4%	0.0	35.0%	0.0	**0.0%**	0.0	2.6%	0.0	-	3600.0	1.5%	0.0	35.6%	0.0	**0.6%**	0.0	5.2%	0.0
		Average	38.4%	3600.0	21.2%	0.0	44.4%	0.0	**12.8%**	0.0	14.0%	0.0	-	3600.0	21.7%	0.0	44.1%	0.0	**12.2%**	0.0	13.0%	0.0
	28	Best	19.4%	3600.0	1.6%	0.0	39.2%	0.0	**1.1%**	0.0	4.9%	0.0	-	3600.0	3.1%	0.0	41.0%	0.0	**0.9%**	0.0	**0.9%**	0.0
		Average	36.5%	3600.0	16.7%	0.0	49.3%	0.0	15.2%	0.0	**15.0%**	0.0	-	3600.0	17.4%	0.0	48.9%	0.0	14.4%	0.0	**13.5%**	0.0
Subgroup Average			31.6%	3600.0	20.6%	0.0	38.2%	0.0	**11.1%**	0.0	11.5%	0.0	*	3600.0	21.0%	0.0	37.9%	0.0	**10.5%**	0.0	10.7%	0.0
40	24	Best	40.2%	3600.0	15.7%	0.0	20.9%	0.0	**3.9%**	0.0	4.5%	0.0	-	3600.0	2.3%	0.0	50.0%	0.0	**0.0%**	0.0	1.8%	0.0
		Average	46.3%	3600.0	20.7%	0.0	26.2%	0.0	9.6%	0.0	**9.0%**	0.0	-	3600.0	8.6%	0.0	56.0%	0.0	6.2%	0.0	**5.6%**	0.0
	32	Best	52.6%	3600.0	21.3%	0.0	25.2%	0.0	9.0%	0.0	**8.3%**	0.0	-	3600.0	22.4%	0.0	27.4%	0.0	**6.0%**	0.0	**6.0%**	0.0
		Average	57.0%	3600.0	26.5%	0.0	31.8%	0.0	12.8%	0.0	**12.5%**	0.0	-	3600.0	28.5%	0.0	32.7%	0.0	11.8%	0.0	**11.0%**	0.0
	40	Best	57.3%	3600.0	26.1%	0.0	31.7%	0.0	9.1%	0.0	**8.3%**	0.0	-	3600.0	24.4%	0.0	32.6%	0.0	**9.0%**	0.0	10.4%	0.0
		Average	69.2%	3600.0	31.1%	0.0	36.1%	0.0	12.5%	0.0	**12.3%**	0.0	-	3600.0	32.4%	0.0	37.7%	0.0	**13.3%**	0.0	13.6%	0.0
	48	Best	-	3600.0	23.9%	0.0	41.6%	0.0	**9.5%**	0.0	11.6%	0.0	-	3600.0	23.0%	0.0	41.7%	0.0	**10.8%**	0.0	11.6%	0.0
		Average	-	3600.0	29.6%	0.0	45.2%	0.0	**16.6%**	0.0	16.8%	0.0	-	3600.0	29.1%	0.0	45.3%	0.0	16.6%	0.0	**16.1%**	0.0
	56	Best	-	3600.0	16.8%	0.0	44.4%	0.0	**13.2%**	0.0	15.5%	0.0	-	3600.0	18.4%	0.0	44.8%	0.0	**13.8%**	0.0	14.8%	0.0
		Average	-	3600.0	23.3%	0.0	48.3%	0.0	19.7%	0.0	**19.3%**	0.0	-	3600.0	23.3%	0.0	48.3%	0.0	18.7%	0.0	**18.1%**	0.0
Subgroup Average			*	3600.0	26.2%	0.0	37.5%	0.0	14.2%	0.0	**14.0%**	0.0	*	3600.0	24.4%	0.0	44.0%	0.0	13.3%	0.0	**12.9%**	0.0
60	36	Best	-	3600.0	16.9%	0.0	20.9%	0.0	5.5%	0.0	**4.5%**	0.0	-	3600.0	17.8%	0.0	21.4%	0.0	5.9%	0.0	**3.4%**	0.0
		Average	-	3600.0	22.8%	0.0	26.7%	0.0	8.6%	0.0	**8.4%**	0.0	-	3600.0	22.7%	0.0	26.6%	0.0	8.4%	0.0	**7.6%**	0.0
	48	Best	-	3600.0	26.0%	0.0	30.4%	0.0	**9.6%**	0.0	10.7%	0.0	-	3600.0	26.5%	0.0	30.6%	0.0	9.4%	0.0	**8.5%**	0.0
		Average	-	3600.0	29.9%	0.0	33.0%	0.0	**12.0%**	0.0	12.8%	0.0	-	3600.0	29.6%	0.0	32.9%	0.0	**11.2%**	0.0	12.0%	0.0
	60	Best	-	3600.0	26.6%	0.0	35.3%	0.0	14.2%	0.0	**13.9%**	0.0	-	3600.0	28.0%	0.0	36.1%	0.0	13.3%	0.0	**12.9%**	0.0
		Average	-	3600.0	33.5%	0.0	40.6%	0.0	**18.2%**	0.0	18.3%	0.0	-	3600.0	33.6%	0.0	40.7%	0.0	**16.9%**	0.0	17.0%	0.0
	72	Best	-	3600.0	25.7%	0.0	42.6%	0.0	15.5%	0.0	**14.8%**	0.0	-	3600.0	26.0%	0.0	42.8%	0.0	**13.5%**	0.0	**13.5%**	0.0
		Average	-	3600.0	30.6%	0.0	46.2%	0.0	19.1%	0.0	**18.9%**	0.0	-	3600.0	30.5%	0.0	46.0%	0.0	18.2%	0.0	**17.6%**	0.0
	84	Best	-	3600.0	20.3%	0.0	44.0%	0.0	19.3%	0.0	**18.4%**	0.0	-	3600.0	20.2%	0.0	44.4%	0.0	17.8%	0.0	**17.4%**	0.0
		Average	-	3600.0	26.0%	0.0	50.4%	0.0	21.9%	0.0	**21.4%**	0.0	-	3600.0	25.9%	0.0	50.3%	0.0	21.4%	0.0	**20.5%**	0.0
Subgroup Average			*	3600.0	28.6%	0.0	39.4%	0.0	**16.0%**	0.0	**16.0%**	0.0	*	3600.0	28.5%	0.0	39.3%	0.0	15.2%	0.0	**14.8%**	0.0

Table 3.3 compares the lower bounds obtained for each method. As it is possible to see, not only the hybrid approach achieves the best upper bounds, but also it systematically generates the best lower bounds, proving to be a very efficient and complete method. It is important to point out, that the Hybrid Lagrangian approach significantly improves the linear-relaxation bounds (column LR), even having polynomially solvable subproblems. This occurs because each time we improve the upper bound (makespan) we can transform the original problem, reducing the number of variables, allowing to improve the previous linear bound.

Table 3.3: Computational results for Lower Bounds.

	Group 1 - Processing time [1, 10]				Group 2 - Processing time [10, 100]			
n	LR	LB_1	LB_2	LB_{HgR}	LR	LB_1	LB_2	LB_{HgR}
5	23.7	29.3	33.6	**34.7**	232.8	288.9	337.4	**347.0**
10	39.4	60.9	66.2	**70.0**	379.1	635.9	667.6	**706.9**
20	70.7	119.7	140.2	**143.4**	705.7	1203.0	1412.7	**1442.4**
40	132.9	225.5	293.5	**294.9**	-	2253.2	2871.8	**2883.2**
60	193.9	338.0	**444.3**	**444.3**	-	3373.5	**4452.3**	**4452.3**

Chapter 4

Generalization for parallel-docks CDC

4.1 Introduction

In this section, we present a time-indexed mixed integer formulation for $F2(P)|CD|C_{max}$ based on the proposal of Cota et al. (2016). The cross-docking problem is modeled as a hybrid two-stage flow shop scheduling problem with identical machines and cross-docking constraints, with the objective of minimizing the makespan. The problem is denoted as $F2(P)|CD|C_{max}$, since $F2|CD|C_{max}$ is strongly NP-hard, as showed by Chen and Lee (2009), it is not difficult to see that $F2(P)|CD|C_{max}$ is also strongly NP-hard.

4.2 Time-indexed mixed integer linear programming model for $F2(P)|CD|C_{max}$

For the parallel-docks formulation, we use the following notation:

Input Parameters

- m_1: number of parallel processors in stage 1.
- m_2: number of parallel processors in stage 2.
- n_1: number of jobs (inbound trucks) in stage 1.
- n_2: number of jobs (outbound trucks) in stage 2.
- p_j: processing time of job j.

- T: Time horizon considered initially. In practice, we used $T = \sum_{j \in J_1} p_j + \sum_{j \in J_2} p_j$.

Sets

- $T = \{0, ..., T\}$, set of discrete periods considered.
- $J_1 = \{1, 2, ..., n_1\}$, set of jobs in stage 1.
- $J_2 = \{1, 2, ..., n_2\}$, set of jobs in stage 2.
- S_j: a set of precedent subset jobs of J_1 corresponding to job $j \in J_2$, $S_j \subset S = \{S_1, ..., S_{n_2}\}$.

Decision variables

- C_{max}: makespan.
- $x_{jt} = 1$, if job j starts to be processed at time t, 0, otherwise.

The multiple dock formulation is defined as:

$$Minimize\ C_{max} \tag{4.1}$$

subject to

$$\sum_{t=0}^{T-p_j} x_{jt} = 1, \quad \forall j \in J_1, \tag{4.2}$$

$$\sum_{t=0}^{T-p_j} x_{jt} = 1, \quad \forall j \in J_2, \tag{4.3}$$

$$\sum_{j \in J_1} \sum_{s=max(0;t-p_j+1)}^{t} x_{js} \leq m_1, \quad \forall t \in T, \tag{4.4}$$

$$\sum_{j \in J_2} \sum_{s=max(0;t-p_j+1)}^{t} x_{js} \leq m_2, \quad \forall t \in T, \tag{4.5}$$

$$\sum_{t=0}^{T-p_j} t x_{jt} \geq \sum_{t=0}^{T} (t + p_i) x_{it}, \quad \forall j \in J_2, \forall i \in S_j, \tag{4.6}$$

$$C_{max} \geq p_j + \sum_{t=0}^{T-p_j} t x_{jt}, \quad \forall j \in J_2, \tag{4.7}$$

$$x_{jt} \in \{0, 1\}, \quad \forall j \in J_1 \wedge \forall j \in J_2, \forall t \in T, \tag{4.8}$$

$$C_{max} \geq 0. \tag{4.9}$$

The objective function remains as the minimization of the makespan. Constraints (4.2) and (4.3) ensure that in every stage, each job should start its processing on one, and only one, time slot within the time horizon T. Constraints set (4.4) and (4.5) indicates that the number of jobs that can be process simultaneously is less than or equal to the number of existing parallel processors in the stage considered. The constraint set (4.6) represents the cross-docking constraints, in which the release date of each job $j \in J^2$, must be greater than or equal to the completion date of each task belonging to its precedence set. Besides that, the constraint set ensures that each job $j \in J^2$, starts processing only after its release date. Constraints (4.7) compute the makespan, analyzing the maximum completion time. Constraints (4.8) and (4.9) specify the domains of each decision variable.

4.3 Lagrangian Relaxation for $F2(P)|CD|C_{max}$

Let $L_{MD}(\lambda)$ be the relaxation of multiple dock formulation, when the cross-docking constraints (4.6) are dualized and its violation penalized in the objective function. We define the subproblem $L_{MD}(\lambda)$ as:

$$L_{MD}(\lambda) = Minimize\ C_{max} - \sum_{j \in J_2} \sum_{i \in S_j} \lambda_{ij} \Big(\sum_{t=0}^{T-p_j} t x_{jt} - \sum_{t=0}^{T} (t + p_i) x_{it} \Big) \tag{4.10}$$

s.t. (4.2) – (4.5), (4.7) – (4.9), $\lambda_{ij} \geq 0$.

Just as we did for the 2-dock case, we are able to uncouple the problem into two new scheduling problems, one for each stage.

Subproblem 1

$$L'_{MD}(\lambda) = Min \sum_{j \in J_2} \sum_{i \in S_j} \lambda_{ij} \sum_{t=0}^{T} (t + p_i) x_{it} \tag{4.11}$$

subject to

$$\sum_{t=0}^{T-p_j} x_{jt} = 1, \qquad \forall j \in J_1, \tag{4.12}$$

$$\sum_{j \in J_1} \sum_{s=max(0;t-p_j+1)}^{t} x_{js} \leq m_1, \qquad \forall t \in T, \tag{4.13}$$

$$x_{jt} \in \{0, 1\}, \qquad \forall j \in J_1, \forall t \in T, \tag{4.14}$$

$$\lambda_{ij} \geq 0, \qquad \forall j \in J_2, i \in S_j. \tag{4.15}$$

Subproblem 2

$$L''_{MD}(\lambda) = Min\ C_{max} - \sum_{j\in J_2}\sum_{i\in S_j}\lambda_{ij}\sum_{t=0}^{T-p_j} t x_{jt} \tag{4.16}$$

subject to

$$\sum_{t=0}^{T-p_j} x_{jt} = 1, \qquad \forall j \in J_2, \tag{4.17}$$

$$\sum_{j\in J_2}\sum_{s=max(0;t-p_j+1)}^{t} x_{js} \leq m_2, \qquad \forall t \in T, \tag{4.18}$$

$$C_{max} \geq p_j + \sum_{t=0}^{T-p_j} t x_{jt}, \qquad \forall j \in J_2, \tag{4.19}$$

$$x_{jt} \in \{0,1\}, \qquad \forall j \in J_2, \forall t \in T, \tag{4.20}$$

$$\lambda_{ij} \geq 0, \qquad \forall j \in J_2, i \in S_j, \tag{4.21}$$

$$C_{max} \geq 0. \tag{4.22}$$

4.4 Constructive Heuristic for parallel-dock CDC

The first subproblem can be defined as $Pm||\sum C_j W_j$, while the second as $Pm||C_{max} - \sum I_j W_j$, therefore, it is not difficult to see that both subproblem are strongly NP-hard (see Pinedo (2008)). In this case, we focus on improving the problem's upper bounds, thus, in each iteration, we solve the linear relaxations of the subproblems, obtain integer solutions and then compute their Lagrangian multipliers. The logic of the procedure is described in Algorithm 1, maintaining the same reasoning proposed in Section 4.

For comparison purposes, we limited the computational time in 60 seconds, and we compute the LB precisely as proposed by Chen and Song (2009).

4.5 Results for the multi-dock problem

To investigate the performance of the proposed heuristics and the heuristics presented in literature, we generate artificial instances varying the number of jobs and machines, as performed in Chen and Song (2009). The heuristics are coded in C++ and solved using AMPL and CPLEX 12.4[1]. Tests were performed on a single thread in Intel

[1]all instances and codes are available at `https://github.com/GabrielaBragaFonseca/Cross-docking-Problems`

Algorithm 1 Algorithm MD

1: Set Lagrangian multipliers as null.
2: Compute a Lower Bound (LB), as proposed by Chen and Song (2009).
3: Compute an initial Upper Bound (UB).

$$UB = \sum_{i \in J_1} p_i/m_1 - max_{i \in J_1}\, p_i/m_1 + max_{i \in J_1}\, p_i + \\ + \sum_{j \in J_2} p_j/m_2 - max_{j \in J_2}\, p_j/m_2 + max_{j \in J_2}\, p_j.$$

4: **While** Time < TimeLimit
5: Solve the Linear Relaxation.
6: Allocate the job as long as its variable is closer to 1.
7: Calculate the Lagrangian multipliers.
8: Compute the weights β_1 and β_2 for the heuristics H1 and H2 (the ordered weights define the entry list for H1 and H2).
9: Apply Heuristics H1 and H2.
10: Apply a First-Best Local Search.
11: **if** UB has improved
12: Update the UB and add a cut to reduce the time horizon.
13: **end if**
14: **end while**
15: **end algorithm**.

Xeon-Silver 4110 (2.1GHz/8-core/85W) with 64 GB memory and Linux operational system.

4.5.1 Instances Generation

The proposed heuristics H1 and H2 are compared with the best heuristics in Chen and Song (2009), called JRH and JLPTH, and with the CDH heuristic Cota et al. (2016). We consider values of n_1 equal to 20, 30, 40, 50, 60, 70 and 80, and values of n_2 equal to integers in $[0.8n_1, 1.2n_1]$. We examine five groups of instances. The first three groups consider the same number of machines in each stage, $2, 4$, and 10. The last two groups of instances use a random number of machines in each stage, selected from a discrete uniform distribution $U(2, 4)$ and $U(2, 10)$, respectively. Processing times were generated with a discrete uniform distribution $U(10, 100)$. A job $j \in J_1$, $J_1 = \{1, 2, ..., n_1\}$ has a probability of 50% to belong to the set S_j of each job $j \in J_2$, $J_2 = \{1, 2, ..., n_2\}$. We generate 300 instances for each combination of number of jobs and number of machines, resulting in a total of 10.500 instances.

4.5.2 Computational Results

In Table 4.1 we report computational results obtained with heuristics JRH, JLPTH, CDH, H1 and H2 for large instances. We coded the JRH and JLPTH heuristics in C++ following the description found in Chen and Song (2009). For each combination of number of jobs and number of machines and for each heuristic we report best, average and worst percentage Loss over 300 instances, along with standard deviations. In order to evaluate the heuristics proposed we used 'Loss' as the criterion to balance the quality of each heuristic algorithm, Loss = (makespan-lower bound)/lower bound, as defined by Chen and Song (2009).

Table 4.1: Computational results for large instances. The best, the average and the worst percentage Loss values are presented for each instance configuration.

m_i	n_1	n_2		Loss(%)				
				JRH	JLPTH	CDH	H1	H2
2	20	[16,24]	Best	16.92	16.76	17.69	9.02	9.76
			Average	32.34	30.91	35.37	20.93	22.20
			Worst	53.77	51.24	64.08	36.36	35.08
			Std Dev	6.73	6.49	8.67	4.84	4.74
	30	[24,36]	Best	17.83	17.14	21.31	11.62	12.46
			Average	33.94	32.98	37.59	24.25	25.88
			Worst	46.49	45.84	64.72	37.11	36.38
			Std Dev	5.39	5.18	6.96	4.35	4.13
	40	[32,48]	Best	22.80	22.29	24.60	15.54	17.96
			Average	35.29	34.52	38.67	26.75	28.36
			Worst	47.91	45.72	59.40	37.13	39.12
			Std Dev	4.34	4.25	6.41	3.80	3.62
	50	[40,60]	Best	25.58	25.18	23.65	19.44	19.36
			Average	35.77	35.03	38.94	28.01	29.62
			Worst	46.22	45.58	59.36	35.91	38.07
			Std Dev	3.96	3.91	6.28	3.32	3.29
	60	[48,72]	Best	25.93	26.12	26.65	18.83	21.02
			Average	36.25	35.67	38.93	29.20	31.01
			Worst	46.69	45.47	56.3	36.72	38.43
			Std Dev	3.62	3.62	5.33	3.24	3.12
	70	[56,84]	Best	29.25	28.84	28.02	22.52	23.22
			Average	36.52	36.01	39.81	30.35	31.51
			Worst	44.62	44.49	56.35	39.09	39.07
			Std Dev	3.24	3.20	5.30	3.02	2.88
	80	[64,96]	Best	28.84	28.07	28.01	22.46	25.42
			Average	36.52	36.12	39.58	30.99	32.18
			Worst	43.73	43.52	55.55	38.98	39.38
			Std Dev	2.97	2.94	5.13	2.89	2.73
Average 2 machines				35.23	34.46	38.41	27.21	28.68
4	20	[16,24]	Best	20.90	17.36	17.56	11.57	12.99
			Average	39.87	34.38	35.89	24.78	25.31
			Worst	61.61	52.48	59.75	35.31	35.00
			Std Dev	7.27	6.15	7.43	4.65	4.33
	30	[24,36]	Best	26.60	24.04	24.02	17.80	17.47

(Continued on next page)

Table 4.1 (continued)

m_i	n_1	n_2		Loss(%)				
				JRH	JLPTH	CDH	H1	H2
			Average	39.92	36.17	37.29	28.05	28.88
			Worst	51.71	46.49	59.00	36.70	39.66
			Std Dev	5.17	4.54	6.28	3.75	3.67
	40	[32,48]	Best	26.79	24.49	22.00	19.42	20.38
			Average	39.68	36.79	37.64	29.35	30.25
			Worst	50.72	46.13	54.92	38.28	39.28
			Std Dev	4.31	3.87	5.63	3.47	3.16
	50	[40,60]	Best	27.85	26.69	21.9	20.73	21.36
			Average	39.27	36.71	37.66	30.29	31.25
			Worst	49.65	47.95	57.31	39.26	38.59
			Std Dev	3.89	3.68	5.43	3.29	3.06
	60	[48,72]	Best	28.57	27.55	26.52	21.08	24.53
			Average	39.01	36.96	37.7	31.07	32.05
			Worst	49.20	45.64	51.43	39.82	38.62
			Std Dev	3.44	3.35	4.68	2.94	2.83
	70	[56,84]	Best	30.86	30.08	26.71	23.55	24.86
			Average	38.75	36.99	38.21	31.52	32.49
			Worst	47.80	44.42	52.69	38.48	38.67
			Std Dev	3.14	2.90	4.57	2.82	2.62
	80	[64,96]	Best	30.27	29.09	27.63	22.95	26.10
			Average	38.39	36.94	38.33	32.17	33.12
			Worst	44.81	43.51	50.97	38.69	39.38
			Std Dev	2.86	2.77	4.47	2.65	2.60
Average 4 machines				39.27	36.42	37.53	29.60	30.48
10	20	[16,24]	Best	12.54	11.59	5.69	0.00	4.10
			Average	35.38	24.56	28.23	13.50	16.49
			Worst	55.41	45.50	61.03	27.01	27.27
			Std Dev	7.52	5.74	8.44	4.79	4.24
	30	[24,36]	Best	26.91	18.44	18.53	12.24	13.86
			Average	43.46	32.08	34.65	24.14	24.71
			Worst	59.20	45.91	53.51	34.19	35.02
			Std Dev	6.31	5.16	6.64	4.60	4.46
	40	[32,48]	Best	28.06	22.54	27.20	15.96	18.67
			Average	49.17	38.74	40.87	31.79	32.10
			Worst	63.17	48.35	57.14	41.04	40.96
			Std Dev	6.04	4.87	6.21	4.33	4.21
	50	[40,60]	Best	36.50	28.39	27.36	26.04	25.41
			Average	50.62	42.04	42.83	35.96	35.81
			Worst	63.10	51.89	57.84	43.15	41.23
			Std Dev	4.79	3.77	5.27	3.28	3.20
	60	[48,72]	Best	38.27	32.56	30.42	27.74	27.91
			Average	49.40	41.85	41.97	36.70	36.44
			Worst	59.34	48.44	57.24	42.09	41.97
			Std Dev	4.04	3.11	4.79	2.71	2.67
	70	[56,84]	Best	37.66	32.57	29.75	28.49	29.02
			Average	47.62	41.21	41.24	36.53	36.24
			Worst	59.36	46.98	52.51	42.02	41.12
			Std Dev	3.55	2.63	4.57	2.50	2.34
	80	[64,96]	Best	37.61	33.20	30.17	28.94	28.30
			Average	45.92	40.44	40.82	36.34	36.12
			Worst	53.71	45.87	51.81	41.30	41.29

(Continued on next page)

Table 4.1 (continued)

m_i	n_1	n_2		Loss(%)				
				JRH	JLPTH	CDH	H1	H2
			Std Dev	3.11	2.50	4.38	2.46	2.32
Average 10 machines				45.94	37.28	38.66	30.71	31.13
$U(2,4)$	20	[16,24]	Best	4.57	4.25	9.02	3.85	2.32
			Average	31.33	27.94	36.48	18.62	19.36
			Worst	59.86	46.01	86.91	33.61	35.40
			Std Dev	9.89	8.82	14.55	6.75	6.69
	30	[24,36]	Best	9.57	6.79	10.14	4.82	5.28
			Average	32.07	29.70	38.14	21.86	22.68
			Worst	51.89	47.81	85.47	38.32	36.28
			Std Dev	8.26	7.63	13.76	6.60	6.40
	40	[32,48]	Best	10.90	9.54	10.61	3.00	6.57
			Average	32.33	30.58	38.18	23.72	24.67
			Worst	47.42	44.74	79.10	37.64	36.84
			Std Dev	7.81	7.34	13.05	6.52	6.57
	50	[40,60]	Best	13.26	12.21	12.64	8.33	10.21
			Average	32.65	31.27	38.69	25.04	26.04
			Worst	48.53	45.11	76.62	38.96	37.57
			Std Dev	7.70	7.34	12.63	6.53	6.75
	60	[48,72]	Best	11.15	11.04	11.83	8.34	9.44
			Average	32.74	31.51	38.74	25.72	26.84
			Worst	47.14	43.42	81.55	36.36	38.58
			Std Dev	6.97	6.75	12.09	6.17	6.28
	70	[56,84]	Best	16.49	14.56	16.39	9.94	11.67
			Average	32.36	31.40	39.3	26.16	27.32
			Worst	45.60	43.95	77.17	37.74	39.01
			Std Dev	6.81	6.58	11.96	6.01	6.19
	80	[64,96]	Best	15.01	14.13	15.48	11.03	13.05
			Average	32.53	31.67	39.57	26.95	27.71
			Worst	43.92	43.37	77.62	37.52	37.56
			Std Dev	6.57	6.41	12.05	5.94	6.02
Average $U(2,4)$ machines				32.29	30.58	38.44	24.01	24.94
$U(2,10)$	20	[16,24]	Best	3.40	3.40	2.11	0.15	0.46
			Average	27.78	21.25	35.94	13.15	13.77
			Worst	55.42	43.67	103.55	34.38	31.48
			Std Dev	11.45	9.18	22.69	7.57	7.24
	30	[24,36]	Best	4.35	1.97	4.09	1.18	1.18
			Average	31.67	26.05	40.02	19.44	19.43
			Worst	61.61	45.32	97.89	38.78	37.32
			Std Dev	12.13	10.24	21.41	8.95	8.75
	40	[32,48]	Best	7.67	4.04	4.50	2.14	2.72
			Average	33.90	29.08	40.85	23.07	22.96
			Worst	60.57	48.93	96.09	40.26	39.74
			Std Dev	13.05	10.85	19.21	9.63	9.58
	50	[40,60]	Best	8.09	6.88	4.61	3.58	3.77
			Average	33.64	29.66	41.28	24.30	24.33
			Worst	60.51	50.56	100.21	40.83	40.86
			Std Dev	12.11	10.53	18.97	9.46	9.41
	60	[48,72]	Best	8.81	6.33	6.36	4.54	4.45
			Average	20.78	29.63	41.03	24.64	24.87
			Worst	55.67	47.11	98.85	40.11	41.00
			Std Dev	11.55	10.18	18.76	9.38	9.39

(Continued on next page)

Table 4.1 (continued)

m_i	n_1	n_2		Loss(%)				
				JRH	JLPTH	CDH	H1	H2
	70	[56,84]	Best	7.49	6.18	6.63	4.98	5.18
			Average	32.01	29.25	40.85	24.92	25.08
			Worst	51.53	47.36	93.43	40.86	40.08
			Std Dev	11.07	9.87	18.56	9.28	9.28
	80	[64,96]	Best	8.26	7.51	5.94	5.05	6.07
			Average	31.49	29.10	41.10	25.16	25.32
			Worst	50.58	45.83	92.11	41.67	40.27
			Std Dev	10.57	9.63	18.53	9.01	9.10
Average $U(2,10)$ machines				31.93	27.72	40.15	22.10	22.25
Total average				36.93	33.29	38.64	26.73	27.50

Results show that for all instances sizes, the proposed heuristics H1 and H2 performs better than the heuristics JRH, JLPTH and CDH, for all tested cases (Best, Average, Worst and Std Dev). It is noted that the total average Loss value is considerably lower for H1 and H2.

In this experiment, we are comparing two types of methodologies, on the one hand, we have simple polynomial time heuristics. They are very easy to code, and they are very fast. On the other hand, we have a more complicated method, that will undoubtedly need specific expertise to be implemented. Still, this methodology is very fast, and able to present an average of 10% of makespan benefit independently of the instance tested, reaching in some cases near 20%. It is important to quantify that in an annual operation of 250 days, a 10% difference in the makespan could represent a benefit of 25 days per year.

The heuristic H1 presents a better average result than H2. Furthermore, when the number of machines increases (easier instances) the H2 heuristic performs slightly better than H1 (10 machines). However, when the number of machines decreases (more difficult instances) the performance of the H1 becomes better.

Figures 4.1 and 4.2 summarizes the results. As previously mentioned, we can see that H1 and H2 present the lowest values in all cases. In the Figure, it is possible to see that H1 dominates the results having better Loss values in most cases. Another interesting point is that the proposed heuristics H1 and H2 have similar behavior to JRH heuristic, this may be because these heuristics arrange jobs in the second stage in increasing order of ready times.

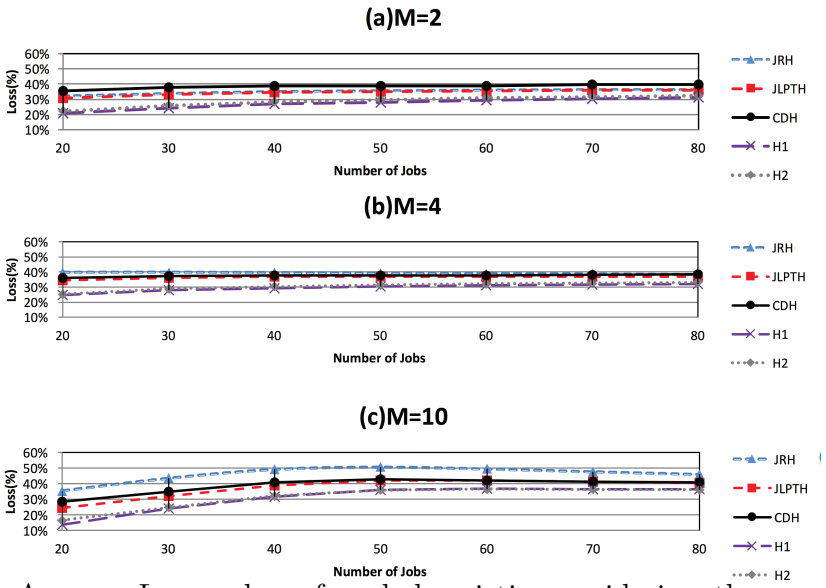


Figure 4.1: Average Loss value of each heuristic considering the same number of machines at each stage: (a) $M = 2$, (b)$M = 4$, (c) $M = 10$.

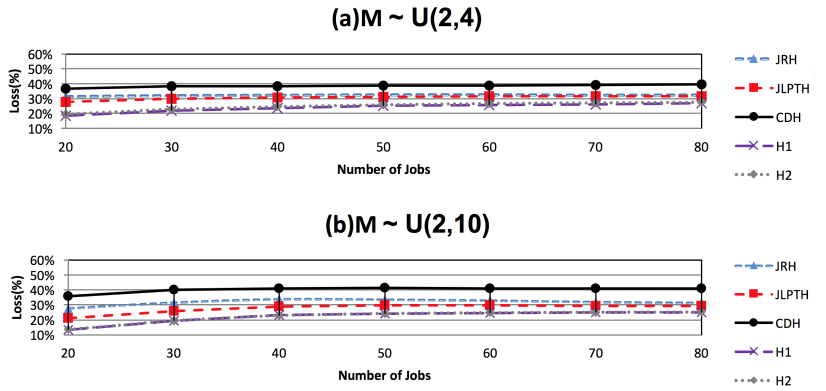


Figure 4.2: Average Loss value of each heuristic considering a random number of machines at each stage: (a) $M \sim U(2, 4)$, (b) $M \sim U(2, 10)$.

Chapter 5

Rescheduling approach to cross-docking truck scheduling problem with truck arrival time uncertainty

5.1 Introduction

Globalization and the rapid growth of e-commerce, now accelerated by the COVID-19 pandemic, have helped change purchase goods and service habits, increasing the complexity of logistics environments. In addition, logistics operations are plagued by uncertainties and disruptions that require a constant revision of strategies and tactics to guarantee the delivery of products within the agreed deadline and in a shorter lead time.

Cross-docking is a logistics solution aiming to move products directly from different suppliers or manufacturers and consolidate them to common final delivery destinations without long-term storage. This strategy allows for fast deliveries and reduced warehousing and transportation costs. Cross-docking Distribution Centers (CDC) need to be flexible and efficient, with quick and real-time responses, minimizing the negative impact of uncertainties throughout the process.

In a CDC, scheduling decisions are particularly relevant to ensure the fast delivery of goods and reduced inventory costs while increasing logistical service, making customers more satisfied. Due to its real-world importance, several truck scheduling problems and procedures have been studied in recent years.

A large number of articles have been published on static scheduling. According to (Gao et al. (2014)), static scheduling assumes that all parameters are deterministic

and that machines and jobs are always available at the beginning of the time horizon. However, in the real world, these assumptions are almost always unfeasible.

Researchers have proposed dynamic scheduling to address the uncertain data problem, of which the most common approach is a predictive-reactive procedure. (Ouelhadj and Petrovic (2009)) explain that in the predictive-reactive procedure, a schedule is initially generated, and over time the schedule is dynamically revised every time a new event happens. For dynamic scheduling, the scheduling problem data (arrival time, machine availability, due date, processing time, among others) are uncertain, which would cause scheduling disruptions.

The problem approached in this Chapter is the multi-dock truck scheduling problem under arrival time uncertainty. The problem is formulated to take into account the uncertainty. In addition, a promised date is established for each outbound truck, and the objective is to minimize the deviation over the scheduled delivery. The proposal aims to integrate existing technologies, such as GPS and algorithms, to define the scheduling of trucks in real-time, generating a more efficient real-world approach.

We analyzed two different problems: the first problem is the problem C_{max}, which the goal is to process all jobs as quickly as possible; the second is the problem WC, which the objective is minimizing the total weighted completion time, taking into account the relevance of each job j to the other jobs in the system. To both problems, we proposed a Rescheduling Approach (RA). The key concept of RA involves rescheduling trucks based on the updated truck arrival date to guarantee the delivery of the goods on the date promised to the customer. The promised date is defined at the end of the day before the operation. During terminal operation and as the arrival dates of the trucks are being updated, the algorithm works to ensure that deliveries are made on promised dates. The performance measures considered in the RA approach are the total tardiness ($\sum T_j$), and the total weighted tardiness ($\sum w_j T_j$) that we believe are correlated to customer satisfaction.

To validate the proposed RA, we proceed with the comparison of three different methodologies, enabling a series of analyses that we believe will help CDC managers deal with the uncertainties generated by delays or advances in truck arrivals, supporting daily dynamic scheduling decisions at a cross-docking center.

The contributions of the proposed study are summarized as follows:

1. An efficient rescheduling approach and a novel algorithm for solving multi-dock truck scheduling problem under truck arrival time uncertainty applied to two different optimization problems (C_{max} and WC);

2. Comparison of three different methodologies to deal with uncertainty;

3. The combination of existing technologies with efficient algorithms applicable to environments with high and low levels of uncertainties achieving good answers;

4. Practical discussion: our methodology can support managers in their daily cross-docking operations, handle dynamic and uncertain data, and solve online problems quickly.

The Chapter 5 is organized as follows. The problem statement is described in Section 5.2, and the mathematical formulation of the problems are described in Section 5.3. In Section 5.4, the details of the proposed RA strategy are presented. Next, the experimental results are discussed in Section 5.5.

5.2 Problem statement

Our work focus is on solving the multi-dock cross-docking scheduling problem with truck arrival time uncertainty. Let us consider a CDC that operates between 8 am and 6 pm, in which each inbound truck has an Estimated Time of Arrival (ETA). The cross-dock manager only knows the ETA and needs to define the unloading scheduling of incoming trucks and, consequently, the loading scheduling of outgoing trucks. The ETA may vary over time, and depending on these values, the discharge sequences at the cross-docking center can be reorganized to reduce the deviation from the scheduled delivery. Our approach aims not to deliver faster, but to guarantee delivery on the date promised to customers. This delivery date is calculated considering the last estimate of the arrival date of the trucks (before the operation day). In practice, real-time data can be collected, for example, through the GPS installed in the trucks. The problem is defining a sequence to unloading the arrival trucks and loading the outgoing trucks at each check time to guarantee the minimum possible delay in the deliveries previously agreed with the customers.

To better understand our proposal, we summarize some important information:

- We assume a cross-docking center that operates from 8 am to 6 pm and a sequencing checkpoint every thirty minutes. Therefore, twenty-one check times are considered ($HT = 21$ check times);
- Each incoming truck has an estimated time of arrival (ETA), defined by the release date (rd_{jh}).
- The arrival dates (rd_{jh}) are updated every thirty minutes (check times) and an evaluation of the scheduling is done to verify the need for rescheduling of jobs (called tolerance τ_1, τ_2).

- The arrival date of some trucks can be postponed without increasing the delay in deliveries. These trucks are called "slack jobs".
- Trucks that cannot have their arrival dates delayed are called critical trucks or "critical jobs". The Critical Path Method (CPM) is used to obtain these jobs.
- Each outbound truck $j \in J_2$ has a due date (d_j), corresponding to the delivery dates promised to customers, defined at the end of the day before the operation. The focus is to guarantee deliveries on the promised dates.
- A new scheduling may be generated to minimize the deviation over the programmed delivery.

It is essential to highlight that it does not justify having the exact method. The dynamic problems are usually very complex to be solved optimally by exact methods in a reasonable amount of time. Besides, as we have uncertainty in the arrival times of the trucks, an optimum result for a specific period does not guarantee an optimal outcome for the subsequent periods. When performing tests with the exact model, we observe that it could generate inefficient sequences compared with the proposed method. According to AL-Behadili (2018), unlike static problems, dynamic ones cannot be solved optimally since the optimal schedule depends on future unpredictable real-time events, which only happen after a schedule has been executed.

To determine the best truck scheduling in each check time, considering the updated arrival dates, we have developed an algorithm (Algorithm 2) that uses the Critical Path Method (CPM) logic (see Pinedo (2012) for more information). In contrast to existing static scheduling strategies, our algorithm adjusts the truck scheduling to fulfill the delivery schedule according to the updated truck arrival time information, moment by moment. We define this as 'Rescheduling Approach' (RA). Two other methodologies are generated to compare the performance of the proposed method. A second methodology, named 'without adjustments' (WA), considered the static case and the most common approach found in the literature, considers the twenty-one check times. Still, it does not adjust the sequences to minimize the deviation over the scheduled delivery. The third methodology, called 'Perfect Information' (PI), is a utopic scenario designed as if it would be possible to predict the exact arrival time of the trucks, see in more detail in subsection 5.5.2).

One of the differentials of our research approach is to consider two different optimization problems, the Problem C_{max}, and the problem WC. For both problems, inbound trucks' arrival time is considered under uncertainty, and the objective is to generate a schedule that respects the dates promised to customers as much as possible.

5.3 Mathematical optimization models

This section presents the problem description and two mathematical formulations for the multi-dock cross-docking scheduling problem with truck arrival time uncertainty.

The CDC scheduling problem that we address in this work is modeled as a hybrid two-stage flow shop scheduling problem with identical machines and cross-docking constraints, as proposed in the Chapter 4; however, we incorporate uncertainty in inbound trucks arrival times. The first stage consists of unloading the incoming trucks and the second stage of loading the trucks that leaving the CDC for delivery. There are n_1 inbound trucks assigned to any of m_1 unloading docks of stage 1. Each inbound truck $j \in J_1$ has an estimated time of arrival, defined as rd_j, and an unloading time p_{1j}. Similarly, there are n_2 outbound trucks assigned to any of m_2 loading docks of stage 2. The processing time of each outbound truck j on stage 2 is defined as p_{2j}. Moreover, each outbound truck needs to wait to process a subset of inbound trucks of stage 1 to begin loading. This subset of J_1 for each $j \in J_2$ is defined by parameter (S_j).

We studied two different problems with different optimization criteria. In the first problem, we aim to minimize the total operation time. The total time of a cross-docking operation is known as makespan, which corresponds to the time interval between the beginning of the unloading procedure and the end of loading the last cargo. A high makespan implies that the goods remain for a long time at the CDC, increasing logistical costs and causing delays in delivery.

In the second problem, the optimization objective is to minimize the total weighted completion time of jobs, which concerns the total duration of the loading and unloading operation at the terminal, considering each customer's relevance and the impact of delays in operations. This measure is crucial in a scenario of uncertainties.

For both formulations, we use the following notation:

Input Parameters

- n_1: number of trucks (jobs) on stage 1.
- n_2: number of trucks (jobs) on stage 2.
- m_1: number of docks (machines) on stage 1.
- m_2: number of docks (machines) on stage 2.
- p_{1j}: processing time of truck j on stage 1.
- p_{2j}: processing time of truck j on stage 2.
- S_j : set of predecessors for each outbound truck. For each $j \in J_2$, S_j is a subset of J_1.

- rd_j: arrival date of the inbound truck j.

- T: time horizon length. It is defined as an upper bound for the completion time of all loading trucks, given by: $T = rd_j + \frac{\sum_{j \in J_1} p_{1j}}{m_1} - \frac{p_1^{max}}{m_1} + p_1^{max} + \frac{\sum_{j \in J_2} p_{2j}}{m_2} - \frac{p_2^{max}}{m_2} + p_2^{max}$, in which p_i^{max} is the largest processing time of the truck on stage i. This equation considers that in the worst case the last truck in the sequence has the largest process time.

- r_j^{min}: minimum date for start the outbound truck $j \in J_2$. It is defined as the maximum between the sum of the processing times of its predecessor divided by the number of unloading docks and the largest processing time of its predecessor ($r_j^{min} = max\{\frac{\sum_{k \in S_k} p_{1k}}{m_1}, max_{k \in S_k}\{p_{1k}\}\}$).

Sets

- $J_1 = \{1, 2, ..., n_1\}$, set of jobs on stage 1.

- $J_2 = \{1, 2, ..., n_2\}$, set of jobs on stage 2.

- $T = \{0, ..., T\}$, set of discrete periods considered.

Decision variables

- $x_{jt} (\forall j \in J_1, t \in \{rd_j, ..., T\})$, binary variable, x_{jt} is equal to 1 if inbound truck j is assigned to time t, and 0 otherwise.

- $y_{jt} (\forall j \in J_2, t \in \{r_j^{min}, ..., T\})$, binary variable, y_{jt} is equal to 1 if outbound truck j is assigned to time t, and 0 otherwise.

The formulations are defined below.

5.3.1 Problem C_{max}

In the Problem C_{max}, we aim to minimize the makespan, stands for, the completion time of the last truck loaded to an unloading dock. We add to the variables mentioned above, the non-negative decision variable C_{max}. The CDC scheduling problem can then be formulated as follows:

$$Minimize\ C_{max} \tag{5.1}$$

subject to:

$$\sum_{t=rd_j}^{T-p_{1j}} x_{jt} = 1, \qquad \forall j \in J_1, \tag{5.2}$$

$$\sum_{t=r_j^{min}}^{T-p_{2j}} y_{jt} = 1, \qquad \forall j \in J_2, \tag{5.3}$$

$$\sum_{j \in J_1} \sum_{s=max(rd_j, t-p_{1j}+1)}^{t} x_{js} \leq m_1, \qquad \forall t \in T, \tag{5.4}$$

$$\sum_{j \in J_2} \sum_{s=max(r_j^{min}, t-p_{2j}+1)}^{t} y_{js} \leq m_2, \qquad \forall t \in T, \tag{5.5}$$

$$\sum_{t=r_j^{min}}^{T-p_{2j}} t y_{jt} \geq \sum_{t=rd_i}^{T-p_{1i}} (t + p_{1i}) x_{it}, \qquad \forall j \in J_2, \forall i \in S_j, \tag{5.6}$$

$$C_{max} \geq p_{2j} + \sum_{t=r_j^{min}}^{T-p_{2j}} t y_{jt}, \qquad \forall j \in J_2, \tag{5.7}$$

$$x_{jt} \in \{0, 1\}, \qquad \forall j \in J_1, t \in \{rd_j, ..., T\}, \tag{5.8}$$

$$y_{jt} \in \{0, 1\}, \qquad \forall j \in J_2, t \in \{r_j^{min}, ..., T\}. \tag{5.9}$$

The objective is to minimize the makespan. Constraint sets (5.2) and (5.3) ensure that in every stage, each job should start its processing on one, and only one, time slot within the time horizon T. Constraints set (5.4) and (5.5) indicate that the number of jobs that can be processed simultaneously is less than or equal to the number of existing docks on each stage (unloading and loading). The constraint set (5.6) represents the precedence constraints, in which the start time of the outbound truck must be greater than or equal to the completion time of each inbound truck belonging to its precedence set. Constraints (5.7) compute the makespan, analyzing the maximum completion time of outbound trucks. Constraints (5.8) and (5.9) specify the domains of each decision variable.

5.3.2 Problem WC

In the WC approach, we focus on working with the importance of each delivery to the end customer. For this, we introduce a weighting factor for each outbound truck, and the objective of the problem is to minimize the total weighted completion time ($w_j C_j$).

The total weighted completion time is a crucial factor in reducing inventory costs and increasing delivery efficiency (Zhan et al. (2020)). For the formulation, consider:

- w_j: weight of each outbound truck $j \in J_2$.
- C_j: completion time of job $j \in J_2$, given by: $C_j = \sum_{j \in J_2} \sum_{t=r_j^{min}}^{T-p_{2j}} (t + p_{2j}) y_{jt}$.

The formulation is:

$$Minimize \sum_{j \in J_2} w_j C_j \tag{5.10}$$

s.t. (5.2) – (5.6), (5.8), and (5.9).

5.4 Rescheduling approach - RA

The RA proposed in this work considers that each inbound truck has an ETA value at the end of the day before the cross-docking operation (rd_{j1}). A first schedule is generated based on this information, and the delivery dates promised to customers (Due Dates, d_j) are defined. For each job $j \in J_2$, we calculate their respective Later Start Date (LSD), defining the jobs with slack or not and the size of the idle spaces. In this way, we define the Critical Path (CP), which comprises not slack jobs. The algorithm's objective is to ensure that deliveries are made on the due dates, d_j (customers delivery dates), minimizing delivery delay. Our method evaluates whether there has been a delay or advance in the trucks' arrival forecast and whether this change impacts the current schedule. Based on this information, the RA rescheduled the jobs to minimize the delivery dates previously agreed with the customers. The proposed rescheduling framework is illustrated in Figure 5.1.

The procedure of the rescheduling receives as input data $(n_1; m_1; rd_{jh}; p_{1j}; n_2; m_2; p_{2j}; S_j)$ and can be summarized as follows.

- **Step 1:** Generate an initial schedule (our methodology applies both to problem C_{max} either WC) and obtain the Due Dates for each job (d_j).
- **Step 2:** Calculate the LSD using CPM, obtaining the values of the slack of the outgoing and incoming trucks.
- **Step 3:** Update trucks' arrival dates (rd_{jh}), considering it's ETA and compute the new completion times C_j.
- **Step 4:** Assess the impact of new dates on the current scheduling.
  We define a tolerance (τ_1, τ_2) for the delay or advance of the trucks proportional

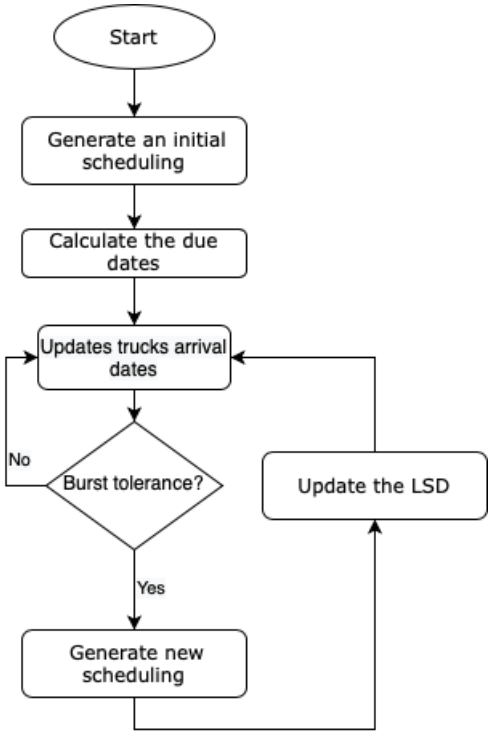


Figure 5.1: Rescheduling scheme.

to the checking instant (h). The closer the trucks' arrival dates, the lower the tolerance. If the truck burst the tolerance, go to **Step 5**; otherwise, go back to **Step 3**.

- **Step 5:** Generate the new schedule to minimize the deviation from the promised date and obtain the objective function values TD and TWT. The jobs are rescheduled taking into account the updated arrival date, respecting as much as possible the sequence defined on the previous day, to keep the minimum deviation from the delivery date previously promised to customers (d_j).
  In Problem C_{max} we obtain $TD = \sum_{j \in J_2} T_j = \sum_{j \in J_2} max\{0, C_j - d_j\}$, where C_j, d_j denote the completion time and due date respectively of job j. In Problem WC we obtain $TWT = \sum_{j \in J_2} w_j T_j$, in which T_j is the tardiness of the job j, and w_j is the weighting factor or importance of customers.

Repeat the above procedure (Steps 2-5) for the duration of the chosen scheduling horizon (in our case, $h = \{1, 2, ..., HT\}$ in which $HT = 21$ check times), to obtain the schedule implemented with this periodic rescheduling scheme. The algorithm for the proposed RA for handling uncertainty in the arrival time is outlined below.

Algorithm 2 Rescheduling Algorithm

Generate an initial scheduling;
Calculate Due Dates (d_j) based on rd_{j1};
Set $h \longleftarrow 1; NTD \longleftarrow 0; TD \longleftarrow 0; TWT \longleftarrow 0$;
While $h \leq HT$
  Update trucks' arrival dates (rd_{jh}) and compute the new completion times (C_j).
  Compute the tolerances (τ_1 and τ_2).
if $rd_{jh} > \tau_1$ or $rd_{jh} < \tau_2$
    Generate the new scheduling to minimize the deviation from the promised date.
    Compute the objective function values (TD and TWT):

$$TD \longleftarrow \sum_{j \in J_2} max\{0, C_j - d_j\};$$

$$TWT \longleftarrow \sum_{j \in J_2} w_j T_j;$$

  end if
for $(j \in J_2)$
if $C_j - d_j > 0$
$NTD \longleftarrow NTD + 1$;
  end if
  end for
    $h \longleftarrow h + 1$;
end while
end algorithm

5.5 Experiments and discussion

This section describes the experiments to test the proposed methodology. First, we present the dataset. Next, two scheduling approaches are compared with the proposed rescheduling method. Finally, computational results are discussed. The impact of the parameters on the performance of the RA is analyzed by conducting a comparative analysis which is in the appendix of this work. It is essential to mention that all instances and codes used in this work are available in [1] to ensure future safe comparisons.

5.5.1 Instances generation

To investigate the methodology's performance, we generate artificial instances varying the number of machines and jobs. Instances are generated for a day of operation (from 8 am till 6 pm) to approximate the methods of a real situation. We examine five groups of instances: the first three groups consider the same number of machines on each stage,

[1] https://github.com/GabrielaBragaFonseca/Cross-docking-Problems

2, 4, and 10. The last two groups of instances use a random number of machines on each stage, selected from a discrete uniform distribution $U(2, 4)$ and $U(2, 10)$, respectively. The number of inbound trucks (n_1) follows an uniform distribution between 2 and 5 times the number of machines on stage 1 ($U(2 * m_1, 5 * m_1)$). The number of outbound trucks (n_2) assumes values equal to integers in $[0.8n_1, 1.2n_1]$. We also consider instances with $n_1 = n_2$, $n_1 < n_2$ and $n_1 > n_2$. Processing times were generated with a discrete uniform distribution $U(10, 100)$.

For the problem WC, we consider different weights for the inbound trucks generated from a discrete uniform distribution of $U(1, 10n_2)$. In the problem C_{max}, weights are considered unitary since, for this problem, the customers are equally important.

Moreover, for each inbound truck $j \in J_1$, we defined the arrival dates (rd_{jh}), based on it's ETA. Considering a cross-docking center that operates from 8 am till 6 pm we generate, for each truck, twenty-one ETAs ($h = \{1, 2, ..., 21\}$, in which, $h = 1$ corresponds to 8 am and $h = 21$ corresponds to 6 pm). The arrival dates are designed as follows. The arrival date for the first check time ($h = 1$, or 8 am) is calculated as $rd_{j1} = Min(Max(N(10, 5), 8), 17)$. To determine the remaining 20 arrival estimates, proceed as follows: if $rd_{jh-1} < (8 + 0.5 * (j - 1))$ then $rd_{jh} = rd_{jh-1}$ else $rd_{jh} = Min(Max(rd_{jh-1} * U(0.95, 1.1), 8), 17.5))$.

In total, there are 45 instances for each group of instances, resulting in 225 cases. The methods are coded in C++ and solved using AMPL and CPLEX 12.4. Tests were performed on a single thread in Intel Xeon-Silver 4110 (2.1GHz/8-core/85W) with 64 GB memory and Linux operational system.

5.5.2 Comparison of methodologies

To evaluate the performance of the RA, we develop and analyze other two different scheduling methodologies, namely, Without Adjustments (WA) and Perfect Information (PI). The methodology with adjustments refers to the RA described in Algorithm 2. The WA and PI methodologies are described below:

WA methodology

In this static case, the algorithm does not reschedule jobs according to the updated arrival date. That is, we defined a fixed sequence of the trucks based on the arrival date for the first check time (rd_{j1}), and given this information, described this sequence as being the same for the 21 check times. Jobs are allocated given this fixed sequence respecting the updated arrival time.

PI methodology

In the PI methodology, we know in advance the information of the 21 check times. Thus, we define the sequence of the trucks considering the correct arrival date. This date is computed as: $Real\ r_d = Max(rd_{jh}, h)$, i.e., the maximum date between the estimated truck arrival date, defined in the problem data set, and the current check time ($h = \{1, 2, ..., 21\}$, in which, $h = 1$ corresponds to 8 am and $h = 21$ corresponds to 6 pm). This is the situation under perfect information.

We already expect that RA will perform better than WA and worst than PI. However, the distance of the results against these extremes does not change anything (WA), and knowing everything a priori (PI) will give us an idea of the performance of RA for the types of instances explored.

5.5.3 Computational results

In Table 5.1 we report computational results obtained to problem C_{max} and problem WC for the three methodologies: WA, RA, and PI. As stated previously in section 5.4, to problem C_{max} the performance criteria considered is the minimization of the Total Tardiness (TD), and to problem WC is the minimization of the Total Weighted Tardiness (TWT). In Table 5.1, column OF_{WA} indicates the TD value for the WA methodology, column OF_{RA} displays the TD value for RA, and column OF_{PI} measures the impact of perfect information on the TD objective function. Similarly, columns OF_{WA}, OF_{RA}, and OF_{PI} display the TWT value for the WA, RA, and PI methodologies, respectively. We use 'Pfm' as the criterion to measure the quality of the results. It is computed as $Pfm = 1 - (OF_x - Min(OF_{WA}, OF_{RA}, OF_{PI}))/OF_x$, where OF_x corresponds to the respective value of the objective function (TD or TWT) for a given methodology (WA, RA or PI). The experimental results are shown in the Table 5.1, which contains 'Pfm' average value and a 95% confidence interval (CI) for both the C_{max} and WC.

The results from Table 5.1 yield several relevant insights for a cross-docking truck scheduling problem with truck arrival time uncertainty. First, the rescheduling strategy, as expected, achieved better results for all instances than the static methodology.

For C_{max} problem, the rescheduling strategy manages to be even better than the 'perfect information' methodology. This result is possible because the RA method is an iterative (approximate) method, executed several times over the horizon of 21 check times. Besides that, the fact that problem C_{max} has a pretty symmetric solution space (all customers have equal weights), the method can escape a local optimal where the PI methodology seems to be trapped. Resulting in a more efficient algorithm (RA works as a heuristic strategy).

Table 5.1: Computational results for five groups of instances. The 'Pfm' average value and confidence interval (CI) are presented for each instance group.

m_i		$Pfm(\%)$					
		C_{max}			WC		
		OF_{WA}	OF_{RA}	OF_{PI}	OF_{WA}	OF_{RA}	OF_{PI}
2	Average	83.12	97.84	91.70	78.28	96.94	94.42
	CI	[77.5,88.8]	[96.3,99.4]	[86.8,96.6]	[71.3,85.3]	[94.7,99.2]	[91.2,97.6]
4	Average	80.54	99.15	93.69	72.20	95.10	96.56
	CI	[74.7,86.4]	[98.5,99.8]	[90.9,96.4]	[67.1,77.3]	[92.9,97.3]	[94.3,98.8]
10	Average	77.02	98.84	94.69	67.71	95.43	96.97
	CI	[71.8,82.2]	[97.9,99.8]	[92.9,96.5]	[63.1,72.4]	[93.7,97.2]	[94.6,99.4]
$U(2,4)$	Average	82.63	97.73	89.86	80.95	95.67	95.98
	CI	[77.0,88.2]	[95.1,100.0]	[84.9,94.8]	[75.6,86.3]	[92.9,98.5]	[93.8,98.1]
$U(2,10)$	Average	79.05	97.75	91.34	71.20	93.08	95.75
	CI	[73.3,84.8]	[96.2,99.3]	[88.2,94.5]	[65.6,76.8]	[90.7,95.5]	[92.7,98.8]

When considering the WC case, where customers have different weights, the rescheduling strategy exhibits good performance, but, as expected, the possibility of knowing in advance the delays allows the PI methodology to set a better schedule. However, it is decisive to point out the quality of the RA results.

It is possible to note that the proposed RA algorithm obtained statistically better results with tighter confidence intervals. It reinforces the assumption that the RA is a very efficient approach.

A second metric is established to investigate the performance of the proposed methodologies concerning the number of trucks delayed. This analysis helps assess the ability of methodologies to ensure that deliveries are made on the date agreed with customers. In Table 5.2 we report computational results obtained to problem C_{max} and WC for the three methodologies: WA, RA, and PI. Again, columns NTD_{WA}, NTD_{RA}, and NTD_{PI} display the number of trucks delayed for each case. 'Pfm' is the criterion to measure the quality of each algorithm, computed as $Pfm = 1 - (N_x - Min(N_{WA}, N_{RA}, N_{PI}))/N_x$, where N_x corresponds to the respective value of the number of trucks delayed for the analyzed methodology (WA, RA or PI). The experimental results are shown in the Table 5.2, which contains 'Pfm' average value and a 95% confidence interval (CI) for both the C_{max} and WC.

Analyzing Table 5.2, with this second metric, we can also observe the excellent performance of the RA approach, significantly better than the option without adjustments and very close to the results obtained by PI methodology. These observations give us an idea of the performance expected of the RA against these extreme situations, and we believe it is a flexible alternative to use the available information to make decisions on the spot in a CDC operation.

Table 5.2: Computational results for five groups of instances. The 'Pfm' average value and confidence interval (CI) are presented for each instance group.

m_i		$Pfm(\%)$					
		C_{max}			WC		
		NTD_{WA}	NTD_{RA}	NTD_{PI}	NTD_{WA}	NTD_{RA}	NTD_{PI}
2	Average	89.49	96.79	93.54	92.62	97.44	99.17
	CI	[83.9,95.0]	[93.8,99.8]	[88.2,98.9]	[88.0,97.3]	[95.3,99.6]	[98.0,100.0]
4	Average	86.73	97.43	96.94	87.06	96.08	98.21
	CI	[83.4,90.0]	[95.8,99.1]	[95.1,98.8]	[83.4,90.7]	[93.7,98.4]	[97.0,99.4]
10	Average	87.68	98.98	97.61	90.33	98.10	98.89
	CI	[84.8,90.6]	[98.1,99.8]	[96.3,98.9]	[87.8,92.9]	[96.7,99.5]	[97.9,99.9]
$U(2,4)$	Average	92.14	98.02	95.05	92.79	98.04	99.11
	CI	[87.9,96.4]	[96.0,100.0]	[90.4,99.7]	[88.8,96.8]	[96.2,99.9]	[98.1,100.0]
$U(2,10)$	Average	88.82	97.99	96.94	90.59	97.36	98.12
	CI	[85.4,92.2]	[96.7,99.2]	[94.0,99.9]	[87.1,94.1]	[95.6,99.1]	[95.5,100.0]

Moreover, plots depict the CPU time for each instance group graphically. Figs. 5.2 and 5.3 show that, in general, the problem C_{max} (graphs on the left) requires more time than the problem WC (charts on the right), although both approaches produce very competitive results in terms of execution time. Notice that algorithms solve the problems quickly and reach 120s only for the 'worst-case' (RA when $m_i = 10$). The speed of the resolution method is essential for decisions at the operational level, as is the case for truck scheduling decisions in cross-docking centers, one more reason that justifies the use of our proposed methodology.

Our results can be summarized in three main findings. First, the Rescheduling Approach is an excellent alternative to the static methodology for the simplicity of their implementation and the quality of their results. Second, due to the existence of symmetry in the solution space of the C_{max} problem, our rescheduling algorithm, which is an approximate iterative method, achieved better results when compared to the PI methodology. Finally, our numerical studies show that the rescheduling approach may be applied to environments with high and low levels of uncertainties.

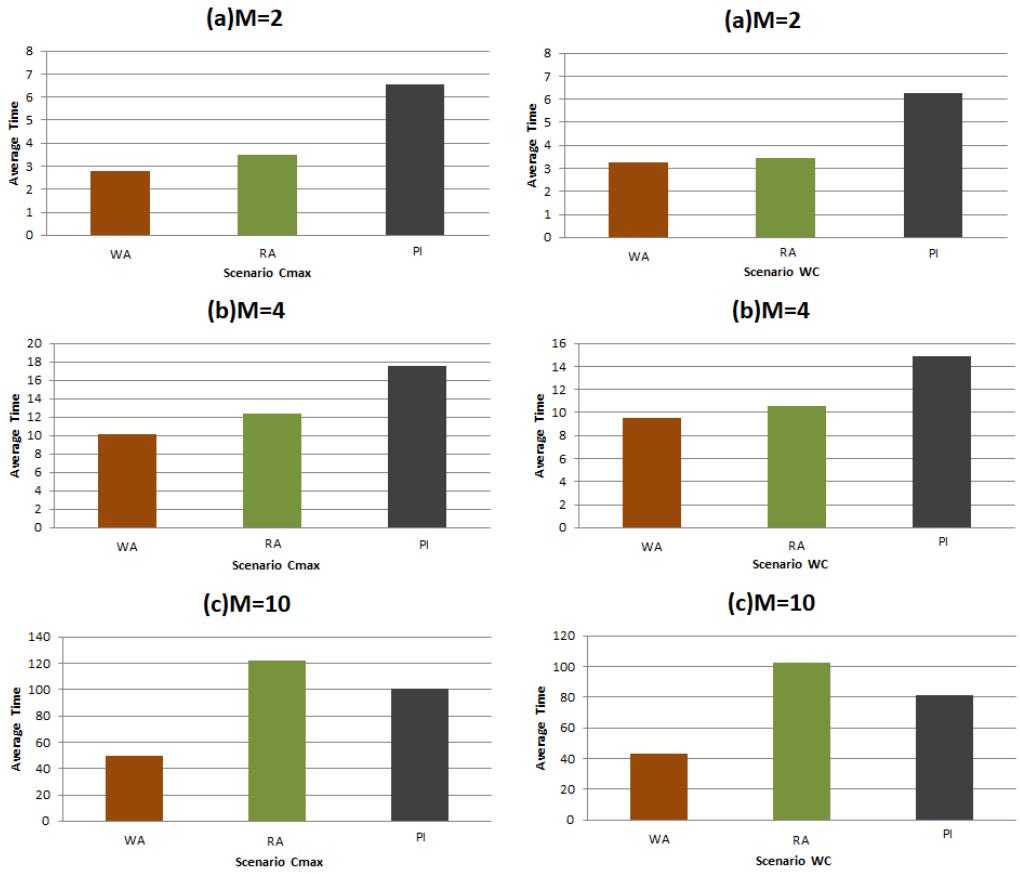


Figure 5.2: Average time in seconds of each instance group considering the same number of machines at each stage: (a) $M = 2$, (b)$M = 4$, (c) $M = 10$.

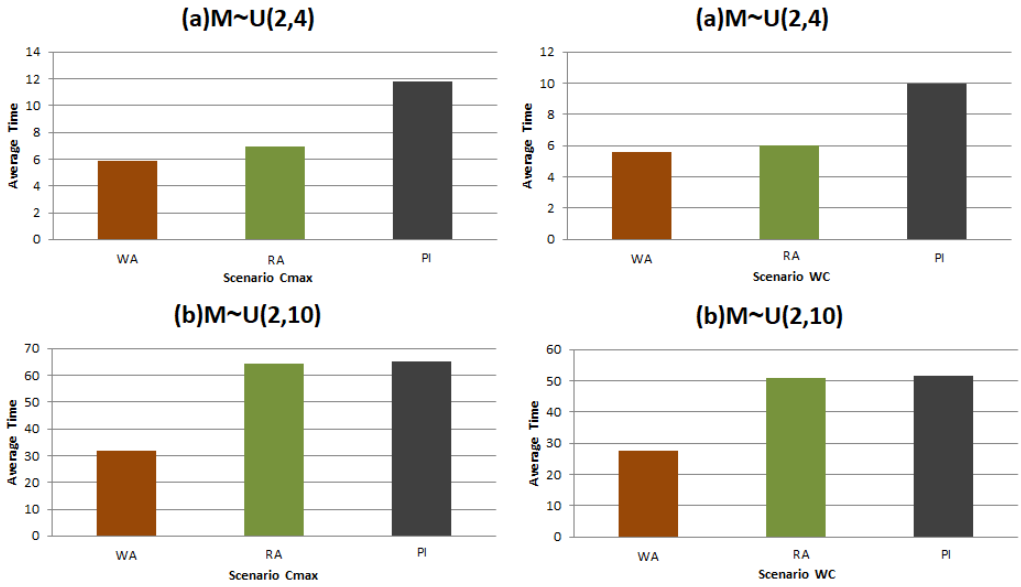


Figure 5.3: Average time in seconds of each instance group considering a random number of machines at each stage: (a) $M \sim U(2, 4)$, (b) $M \sim U(2, 10)$.

Chapter 6

Conclusions

In this work, we propose methods to solve the cross-docking flow shop scheduling problem. Our study developed models, algorithms, and resolution methods for the truck scheduling problem in cross-docking centers, adding uncertainty and delays. Thereby, we hope to fill a critical gap between the current state-of-the-art and the observed industry practice.

Initially, we present a time-indexed formulation and a Hybrid Lagrangian Metaheuristic Framework. As expected, the performance of the MIP is strongly dependent on the instances size, being not able to solve the problem with medium and large dimensions.

For the 2-dock case, even having polynomially solvable Lagrangian subproblems, through a series of cuts, the method improves the model linear-relaxation bound. The Hybrid Lagrangian Metaheuristic shows an efficient performance, obtaining tight bounds in reduced computational time. The heuristics proved to work very well with the Lagrangian approach, finding good solutions, and outperforming previous results. The heuristic H2 gives the best average GAP and Loss results followed by the heuristic H1.

In the generalized version, the subproblems of the Lagrangian relaxation are NP-hard, and we work with their linear relaxations. The Lagrangian heuristics obtain excellent results improving the performance of previous efforts.

Cross-docking is a distribution strategy that enables consolidation shipments to manage better the physical flow of products in a supply chain. In real-world settings, combining existing technologies, like GPS, with efficient algorithms that achieve good answers is crucial for the cross-docking system's daily decisions. Thus, a rescheduling approach to a cross-docking truck scheduling problem with truck arrival time uncertainty is developed to deal with this new, exciting, dynamic environment. The novelty of the proposed Rescheduling Approach is the rescheduling strategy to generate efficient

solutions even when dealing with uncertain arrival dates.

We compared RA against two alternatives (WA and PI) and investigated the impact of the methods on two different problems (C_{max} and WC). From the results, we conclude that RA exhibits excellent features to support managers in their day-to-day operations. It is important to note that despite our analysis focusing on the stochasticity of arrival dates, the same method may be used considering any disruption.

Bibliography

AL-Behadili, M. R. S. (2018). *Robust dynamic and stochastic scheduling in permutation flow shops.* PhD thesis, University of Portsmouth.

Alba, E. (2005). Parallel metaheuristics: a new class of algorithms. *Wiley.com,*, 47.

Alpan, G., Bauchau, S., Larbi, R., and Penz, B. (2008). Optimal operations scheduling in a crossdock with multi strip and multi stack doors. In *38th International conference on computers and industrial engineering - CIE38*, volume 2, pages 1168–1176.

Alpan, G., Larbi, R., and Penz, B. (2011). A bounded dynamic programming approach to schedule operations in a cross docking platform. *Computers & Industrial Engineering,*, 60(3):385–396.

Arabani, A. B., Ghomi, S. F., and Zandieh, M. (2011). Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage. *Expert systems with Applications,*, 38(3):1964–1979.

Araújo, D. P. M. and Melo, B. M. R. (2010). Heurísticas construtivas para o sequenciamento de caminhões em centros de cross-docking. Master's thesis, Universidade Federal de Minas Gerais.

Arroyo, J. E. C., Nunes, G. V. P., and Kamke, E. H. (2009). Iterative local search heuristic for the single machine scheduling problem with sequence dependent setup times and due dates. In *Hybrid Intelligent Systems,*, volume 1, pages 505–510.

Bahiense, L., Maculan, N., and Sagastizabal, C. (2002). The volume algorithm revised: relation with bundle methods. *Mathematical Programming.*, 94.:41–69.

Barahona, F. and Anbil, R. (2000). The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming.*, 87(3):385–399.

Barahona, F. and Ladányi, L. (2006). Branch and cut based on the volume algorithm: Steiner trees in graphs and max-cut. *RAIRO - Operations Research,*, 40:53–73.

Bartholdi, J. and Gue, K. (2002). Reducing labor costs in an ltl crossdocking terminal. *Operations Research,*, 48(6):823–832.

Bartz-Beielstein, T. (2010.). An r package for automatic and interactive tuning of optimization algorithms by sequencial parameter optimization. *arXiv preprint arXiv:1006.4645.*

Bartz-Beielstein, T. and Zaefferer, M. (2012). A gentle introduction to sequential parameter optimization. *Technical Report 2, Bibliothek der Fachhochschule Koeln.*

Belle, J. V., Valckenaers, P., and Cattrysse, D. (2012). Cross-docking: State of the art. *Omega,*, 40:827–846.

Blum, C., Puchinger, J., Raidl, G., and Roli, A. (2011). Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing,*, 11(6):4135–4151.

Boschetti, M. and Maniezzo, V. (2009). Benders decomposition, lagrangean relaxation and metaheuristic design. *Journal of Heuristics,*, 15(3):283–312.

Boysen, N. (2010). Truck scheduling at zero-inventory cross docking terminals. *Computers & Operations Research.*, 37(1):32–41.

Boysen, N. and Fliedner, M. (2010). Cross dock scheduling: Classification, literature review and research agenda. *Omega,*, 38(6):413–422.

Boysen, N., Fliedner, M., and Scholl, A. (2010). Scheduling inbound and outbound trucks at crossdocking terminals. *OR spectrum,*, 32(1):135–161.

Bozer, Y. and Carlo, H. (2008). Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions,*, 40:1007–1018.

Campbell, J. (1994). A survey of network hub location. *Studies in Locational Analysis,*, 6:31–49.

Chen, F. and Lee, C.-Y. (2009). Minimizing the makespan in a two-machine cross-docking flow shop problem. *European Journal of Operational Research,*, 193(1):59–72.

Chen, F. and Song, K. (2009). Minimizing makespan in two-stage hybrid cross docking scheduling problem. *Computers & Operations Research,*, 36(6):2066–2073.

Chen, J. (2007). A hybrid heuristic for the uncapacited single allocation hub location problem. *Omega,*, 35:211–220.

Chen, P., Guo, Y., Lim, A., and Rodrigues, B. (2006). Multiple crossdocks with inventory and time windows. *Computers and Operations Research,*, 33:43–46.

Clausen, J., Cordeau, J., Laporte, G., Wen, M., and J., L. (2009). Vehicle routing scheduling with cross-docking. *Journal of the Operational Research Society.*, 60:1708–1718.

Cota, P. M., Gimenez, B. M., Araújo, D. P., Nogueira, T. H., de Souza, M. C., and Ravetti, M. G. (2016). Time-indexed formulation and polynomial time heuristic for a multi-dock truck scheduling problem in a cross-docking centre. *Computers & Industrial Engineering.*, 95:135–143.

Forger, G. (1995). Ups starts worlds premiere cross-docking operation. *Modern material handling.*, 36(8):36–38.

Fukuda, E. H. (2007). Algoritmo do volume e otimização não diferenciável. Master's thesis, USP.

Gao, H., Kwong, S., Fan, B., and Wang, R. (2014). A hybrid particle-swarm tabu search algorithm for solving job shop scheduling problems. *IEEE transactions on industrial informatics,*, 10(4):2044–2054.

Gaudioso, M., Monaco, M. F., and Sammarra., M. (2020). A lagrangian heuristics for the truck scheduling problem in multi-door, multi-product cross-docking with constant processing time. *Omega,*, In Press.

Gholami, M., Zandieh, M., and Alem-Tabriz, A. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *The International Journal ofAdvanced Manufacturing Technology,*, 42:189–201.

Gue, K. R. (1999). The effects of trailer scheduling on the layout of freight terminals. *Transportation Science.*, 33(4):419–428.

Kim, C., Yang, K. H., and Kim, J. (2008). A strategy for third-party logistics systems: a case analysis using the blue ocean strategy. *Omega,*, 36(4):522–534.

Klose, A. and Drexl, A. (2005). Facility location models for distribution system design. *European Journal of Operational Research,*, 162:4–29.

Konur, D. and Golias, M. M. (2013). Analysis of different approaches to cross-dock truck scheduling with truck arrival time uncertainty. *Computers & Industrial Engineering,*, 65(4):663–672.

Ladier, A. L. and Alpan, G. (2016a). Cross-docking operations: current research versus industry practice. *Omega,*, 62:145–162.

Ladier, A. L. and Alpan, G. (2016b). Robust cross-dock scheduling with time windows. *Computers & Industrial Engineering,*, 99:16–28.

Larbi, R., Alpan, G., Baptiste, P., and Penz, B. (2011). Scheduling cross docking operations under full, partial and no information on inbound arrivals. *Computers & Operations Research,*, 38(6):889–900.

Larbi, R., Alpan, G., and Penz, B. (2009). Scheduling transshipment operations in a multiple inbound and outbound door crossdock. In *39th international conference on computers and industrial engineering - CIE39. Troyes, France (July).*

Lee, K., Lee, Y., and Jung, J. (2008). Positioning of goods in a cross-docking environment. *Computers & Industrial Engineering,*, 54(3):677–689.

Lemaréchal, C. (1975). An extension of davidon methods to non differentiable problems. in: Nondifferentiable optimization. *Springer.*, pages 95–109.

Lemaréchal, C. (2001). Lagrangean relaxation, computational combinatorial. *Springer Verlag.*

Lemaréchal, C. (529-572). Nondifferentiable optimization. *Optimization, Hanbooks in Operations Research.*

Lima, M. F. (2014). O problema de sequenciamento de caminhões numa estação de cross-docking com duas máquinas: Formulação indexada no tempo, relaxação lagrangeana e geração de colunas. Master's thesis, Universidade Federal de Minas Gerais.

Lira, E. G. (2013). Um algoritmo iterated greedy para o problema de sequenciamento de

caminhões em centros de cross-docking. Master's thesis, Universidade Federal de Minas Gerais.

McWilliams, D. L. (2010). Iterative improvement to solve the parcel hub scheduling problem. *Computers & Operations Research,*, 59(1):136–144.

Miao, Z., Lim, A., and Ma, H. (2009). Truck dock assignment problem with operational time constraint within crossdocks. *European journal of operational research.*, 192(1):105–115.

Nawaz, M., Enscore, J., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega,*, 11(1):91–95.

Nie, L., Gao, L., Li, P., and Shao, X. (2013). Reactive scheduling in a job shop where jobs arrive over time. *Computers & Industrial Engineering,*, 66:389–405.

Nogueira, T. H. (2014). *Single machine scheduling problems with sequence-dependent setup times.* PhD thesis, Universidade Federal de Minas Gerais.

Nogueira, T. H., Coutinho, F. P., Ribeiro, R. P., and G., R. M. (2020). Parallel-machine scheduling methodology for a multi-dock truck sequencing problem in a cross-docking center. *Computers & Industrial Engineering,*, 143(106391.).

Oh, Y., Hwang, H., Chab, C., and Lee, S. (2006). A dock-door assignment problem for the korean mail distribution center. *Computers & Industrial Engineering,*, 51(2):288–296.

Ouelhadj, D. and Petrovic, S. (2009). A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling,*, 12(4):417–431.

Paula, M., Mateus, G., and Ravetti, M. (2010). A non-delayed relax-and-cut algorithm for scheduling problems with parallel machines, due dates and sequence-dependent setup times. *Computers & Operations Research,*, 37(5):938–949.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems.* Springer.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems.* Springer Science & Business Media.

Pirkwieser, S., Raidl, G. R., and Puchinger, J. (2007). Combining lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. in: Evolutionary computation in combinatorial optimization. *Springer.*, pages 176–187.

Potts, C. and Strusevich, V. (2009). Fifty years of scheduling: A survey of milestones. *Journal of the Operational Research Society,*, 60:S41–S68.

Puchinger, J. and Raidl, G. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. in: Artificial intelligence and knowledge engineering applications: a bioinspired approach. *Springer.*, pages 41–53.

Smith, W. E. (1956). Varius optimizers for single-stage production. *Naval Res. Logist.*, 3:59–66.

Stalk, G., Evans, P., and Shulman, L. E. (1991). Competing on capabilities: the new rules of corporate strategy. *Harvard business review,*, 70(2):57–69.

Stutzle, T. (1998). *Local search algorithms for combinatorial problems.* PhD thesis, Darmstadt University of Technology.

Tsui, L. and Chang, C. (1990). A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers & Industrial Engineering,*, 19:309–312.

Tsui, L. and Chang, C. (1992). An optimal solution to a dock door assignment problem. *Computers & Industrial Engineering,*, 23:283–286.

Vahdani, B. and Zandieh, M. (2010). Scheduling trucks in cross-docking systems: Robust metaheuristics. *Computers & Operations Research,*, 58(1):12–24.

Vieira, G. E., Herrmann, J. W., and Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling,*, 6:39–62.

Witt, C. E. (1998). Crossdocking: concepts demand choice. *Material Handling Engineering,*, 53(7):44–49.

Wolfe, P. (1975). A method of conjugate subgradients for minimizing nondifferentiable functions. in: Nondifferentiable optimization. *Springer,*, pages 145–173.

Zhan, H., Zhao-hong, J., and Li, K. (2020). Ant colony optimization algorithm for total weighted completion time minimization on non-identical batch machines. *Computers & Operations Research,*, 117.

Appendix A

Resumo Estendido

A.1 Scheduling

Segundo Pinedo (2008), scheduling é um processo de tomada de decisão que desempenha um papel crucial nas indústrias de manufatura e serviços. Problemas de sequenciamento (*Scheduling Problems*) lidam com a atribuição de recursos a tarefas ao longo do tempo. Os recursos e tarefas em uma organização podem assumir muitas formas diferentes. Os recursos podem ser máquinas em uma fábrica, pistas em um aeroporto, equipes em um canteiro de obras, docas em um centro de distribuição, dentre outros. As tarefas podem ser operações em um processo de produção, decolagens e aterrissagens em um aeroporto, estágios em um projeto de construção, caminhões a serem carregados e descarregados em um centro de distribuição, dentre outros exemplos. Cada tarefa pode ter um certo nível de prioridade, uma data de início e uma data de conclusão. Os objetivos também podem assumir muitas formas diferentes. O objetivo do sequenciamento pode ser, por exemplo, a minimização da data de conclusão da última tarefa ou pode ser a minimização do número de tarefas concluídas após suas respectivas datas de conclusão, dentre outros (para mais exemplos de possíveis funções objetivos, ver Pinedo (2008)).

Um sequenciamento eficaz permite não só a sobrevivência da empresa no mercado mas também ganho de vantagem competitiva. Em consequência, as empresas precisam cumprir as datas de entrega que foram prometidas, caso contrário um atraso pode resultar em insatisfação ou perda significativa de clientes. Elas também tem que usar os recursos disponíveis (máquinas, pessoas, equipamentos) de uma maneira eficiente, visando eliminação de disperdícios e perdas (materiais, tempo, espaço, estoques).

O tema sequenciamento começou a ser levado a sério na manufatura no início do século passado com o trabalho de Henry Gantt e outros pioneiros (Pinedo (2008)). No entanto, foram necessários muitos anos para que as primeiras publicações sobre

sequenciamento aparecessem na literatura da engenharia de produção. Algumas das primeiras publicações apareceram no início dos anos 50. Durante os anos sessenta uma quantidade significativa de trabalhos foram feitos em programação dinâmica e em programação inteira. Depois do famoso artigo de Richard Karp sobre a teoria da complexidade, a pesquisa na década de setenta enfocou principalmente a hierarquia de complexidade dos problemas de programação. Nos anos 80, as pesquisas na academia e na indústria tomaram várias direções diferentes, com uma atenção cada vez maior dada aos problemas de programação estocástica.

Os problemas de sequenciamento podem ser encontrados nas mais diversas áreas, desde o planejamento e programação da produção à problemas de bioquímica. Um dos problemas de sequenciamento que tem sido extensivamente estudado durante as últimas décadas é o Problema de *Flow Shop* (FSP). Em um FSP, um conjunto de m máquinas (recursos) tem que processar um conjunto de n *jobs* (tarefas). Todos os *jobs* tem a mesma sequência de operações sobre as máquinas. O FSP tem aplicações em diferentes setores da indústria, entre outros: metalúrgico, químico, têxtil, siderúrgico, etc. Mais recentemente, os problemas de sequenciamento *flow shop* estão sendo aplicados na gestão efetiva da cadeia de suprimentos e logística, especialmente no gerenciamento de centros de *cross-docking*.

A.2 Cross-docking

Centros de *Cross-docking* (CCDs) são pontos intermediários na rede de suprimentos em que se realiza o transbordo de cargas, sem intenção de estocagem, recebendo caminhões com cargas completas de diversos pontos de fornecimento. Dentro dos CCDs, as cargas são descarregadas dos caminhões de chegada, separadas, classificadas, despachadas e diretamente recarregadas em caminhões de saída, de acordo com os pedidos específicos dos clientes. O estoque é reduzido ao mínimo, já que normalmente as mercadorias não permanecem mais do que 24 horas dentro do CCD.

Uma representação esquemática dos processos em um CCD é ilustrada na Figura A.1. O CCD opera recebendo caminhões com cargas de diversos pontos de fornecimento, cada um dos veículos é recebido em uma doca de entrada específica. Dentro do centro, as cargas são descarregadas, separadas, classificadas, combinadas e recarregadas em caminhões de saída, de acordo com os pedidos específicos dos clientes. Os caminhões então deixam o CCD com cargas combinadas, composta por produtos de diversos fornecedores, dedicadas a um cliente ou destino específico.

Os benefícios do *cross-docking* são muitos: redução de custos (custos de armazenamento, manutenção de estoque, manuseio e mão de obra), prazos de entrega mais

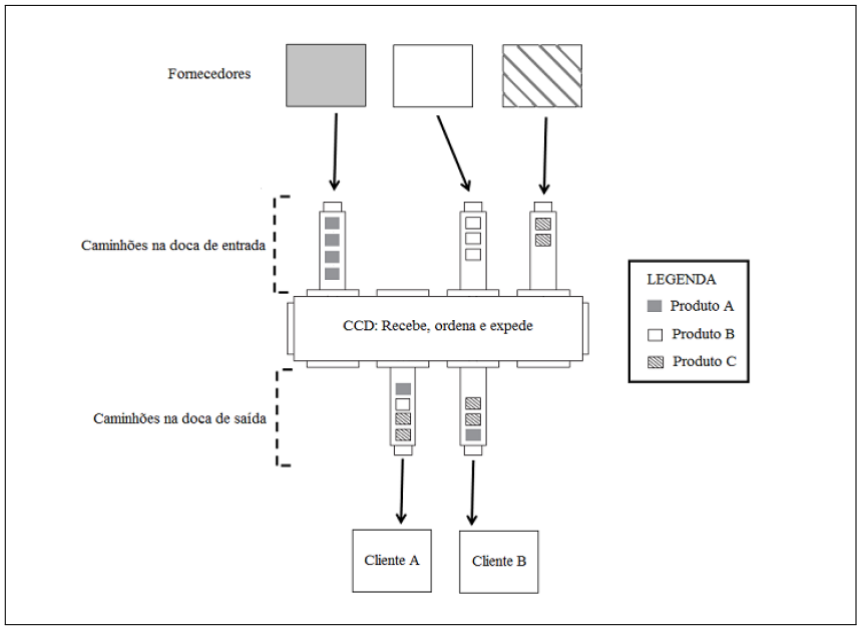


Figure A.1: Rede de distribuição com *Cross-docking*. - Fonte: Lira (2013)

curtos, melhor atendimento ao cliente e satisfação do cliente, redução do espaço de armazenamento, rotatividade de estoques, risco reduzido de perdas e danos. No entanto, processos de transbordo eficientes e planejamento cuidadoso das operações tornam-se indispensáveis dentro de um CCD, onde os fluxos de entrada e saída precisam ser sincronizados para manter o armazenamento do centro o mais baixo possível e para aumentar a confiabilidade das entregas.

Cross-docking é uma técnica de logística amplamente difundida em todo o mundo. Várias empresas bem conhecidas, como redes de varejo (WalMart, Stalk et al. (1991)), empresas de entrega (UPS, Forger (1995)), fabricantes de automóveis (Toyota, Witt (1998)) e provedores logísticos ganharam vantagem competitiva considerável com a utilização de CCDs. Ladier and Alpan (2016a) apresentam uma pesquisa discutindo práticas da indústria e caracterização de problemas nos CCDs.

Existem diferentes abordagens a respeito dos CCDs com enfoques nas diversas etapas do processo logístico. Boysen (2010) e Belle et al. (2012) apresentam uma revisão dos trabalhos presentes na literatura que possuem o tema *cross-docking* como foco principal. Segundo Boysen (2010) problemas de decisão em centros de *cross-docking* podem ser alocados de acordo com a seguinte classificação: localização dos CCDs, *layout*, roteamento de veículos, atribuição de docas, sequenciamento de caminhões e sequenciamento interno de recursos. O problema tratado nesta pesquisa é o sequenciamento de caminhões, onde o objetivo é decidir onde e quando os caminhões devem ser processados.

A.3 Motivação

O atual ambiente de mercado, caracterizado por uma concorrência cada vez mais acirrada, globalização da economia e uma acelerada revolução tecnológica, tem levado as empresas a melhorarem seus sistemas logísticos, de distribuição e produção. Além disso, o aumento populacional e do comércio eletrônico vem exigindo soluções logísticas mais eficientes e eficazes. *Cross-docking* é uma importante técnica logística capaz de reduzir os custos de estocagem enquanto aumenta o fluxo de mercadorias, resultando em uma cadeia de suprimentos muito eficiente.

A escolha do tema de pesquisa foi motivada pelo impacto econômico dos problemas de sequenciamento na cadeia de suprimentos e pela sua aplicabilidade em ambientes industriais e de serviços. Um centro de *cross-docking* foi tomado como base devido à sua importância logística para empresas de diferentes setores. Em comparação com os centros tradicionais de distribuição, um CCD é gerenciado com o mínimo de manuseio e com pouco ou nenhum armazenamento entre o descarregamento e o carregamento de mercadorias. Esta prática pode servir objetivos diferentes: consolidação de cargas, *lead time* de entrega mais curto e principalmente redução de custos, justificando o desenvolvimento de estudos neste campo.

Centros de *cross-docking* levantam inúmeras questões de otimização, sejam elas estratégicas, táticas ou operacionais. Em um CCD, as decisões de sequenciamento são particularmente importantes para garantir uma entrega rápida e pontual. Devido à sua importância no mundo real, vários trabalhos e procedimentos de sequenciamento de caminhões foram introduzidos durante os últimos anos, tratando configurações específicas de *cross-docking*.

Chen and Lee (2009) estudaram o problema de *flow shop cross-docking* com duas máquinas, no qual um *job* na segunda máquina somente pode ser processado após a conclusão dos seus *jobs* precedentes na primeira máquina, o objetivo é minimizar o *makespan*. Os autores mostram que o problema é fortemente NP-difícil e desenvolvem um algoritmo de aproximação polinomial e um algoritmo *branch-and-bound*. Chen and Song (2009) estendem o problema de Chen and Lee (2009) para o problema de *cross-docking* híbrido de dois estágios, considerando vários processadores paralelos (múltiplas docas) por estágio (entrada e saída), permitindo operações simultâneas de carregamento e descarregamento. Eles propõem um modelo de programação inteira mista e quatro heurísticas construtivas, baseadas na regra de *Johnson*, para investigar o desempenho de instâncias de escala moderada e grande. Cota et al. (2016) lidam com o problema de decisão operacional de sequenciar os caminhões em várias docas de entrada e saída, ou seja, eles consideram a mesma configuração que Chen and Song (2009). Uma formulação de programação linear inteira mista indexada no tempo e

uma heurística polinomial construtiva são propostas e testadas em médias e grandes instâncias.

Seguindo a linha dos trabalhos anteriormente citados, considerando uma formulação indexada no tempo, este trabalho propõe métodos eficientes para resolver o problema de sequenciamento de caminhões em um centro de *cross-docking*. Este problema é fortemente NP-difícil (Chen and Song (2009)) justificando a relevância do desenvolvimento de métodos e técnicas capazes de resolver o problema para instâncias médias e grandes.

A.4 Definição do Problema

Considere um centro de *cross-docking* no qual chegam caminhões, cada um carregado com produtos que são demandados por um ou vários clientes, deve-se descarregar os produtos desses caminhões em uma das docas de entrada e carregá-los em caminhões de saída, em uma das docas de saída, estes caminhões são responsáveis por carregar vários tipos de produtos para destinos específicos. Cada caminhão de saída só pode deixar o centro de *cross-docking* se todos os produtos necessários para aquele caminhão já tiverem sido descarregados pelos caminhões de chegada correspondentes em algum momento. O objetivo do problema envolve o sequenciamento dos caminhões de entrada e saída. O problema é ilustrado na Figura A.2 a seguir.

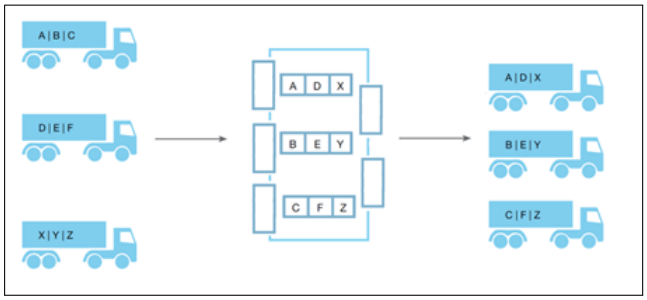


Figure A.2: Exemplo do Sistema de *Cross-docking*.

Este problema foi modelado inicialmente a partir da proposta de Chen and Lee (2009) considerando uma formulação de programação inteira indexada no tempo para o caso com duas docas ($F2|CD|C_{max}$), ou seja, um problema de *flow shop* com duas máquinas, com restrições de *cross-docking* (restrições de precedência), no qual a função objetivo é minimizar a data de conclusão do último caminhão processado pela máquina 2 (*makespan*). Nessa abordagem, consideramos a existência de duas docas no centro, uma doca dedicada a descarregar os caminhões e outra para carregamento, máquina 1 (M1) e máquina 2 (M2), respectivamente.

Posteriormente desenvolvemos uma formulação inteira mista indexada no tempo para o problema com múltiplas docas ($F2(P)|CD|C_{max}$), proposto inicialmente por Chen and Song (2009). O problema foi modelado de forma análoga a um problema de *flow shop* híbrido de dois estágios com máquinas idênticas e paralelas e com restrições que determinam um conjunto de caminhões precedentes para os caminhões do segundo estágio, chamadas de restrições de *cross-docking*. Assim, as máquinas paralelas são análogas às docas de entrada e saída dos caminhões no CCD e os *jobs* são associados aos caminhões. Os tempos de processamento dos *jobs* no primeiro e no segundo estágio correspondem às atividades de descarregamento e carregamento dos caminhões, respectivamente.

Com o objetivo de desenvolver novas perspectivas de pesquisa em relação às operações de *cross-docking* e reduzir o *gap* entre a pesquisa acadêmica e as necessidades industriais, muito bem destacado por Ladier and Alpan (2016a), desenvolvemos uma abordagem de resequenciamento que pode resolver de forma eficiente o problema de sequenciamento de caminhões em um centro de *cross-docking* com incerteza no horário de chegada.

A.5 Contribuições

A fim de compreender melhor a estrutura dos problemas de sequenciamento iniciamos nossas pesquisas considerando primeiramente o problema com duas docas (máquinas), denotado por $F2\big|CD\big|C_{max}$ (NP-difícil, Chen and Lee (2009)). Posteriormente estendemos os estudos para o caso genérico com múltiplas docas, denotado por $F2(P)|CD|C_{max}$. Além de estudar e propor os modelos matemáticos para cada caso, desenvolvemos também uma metaheurística híbrida Lagrangiana. O algoritmo proposto usa as informações obtidas dos multiplicadores de Lagrange para construir soluções viáveis através da heurística NEH, proposta por Nawaz et al. (1983), e executa o procedimento de pesquisa por meio de uma Busca Local estruturada. Os subproblemas Lagrangianos são resolvidos em tempo polinomial, porém a cada iteração, se o *makespan* é melhorado, a formulação indexada no tempo muda e, com isso, melhoramos seu limite inferior linear. Como resultado, podemos provar a otimalidade em vários casos e gerar pequenos *gaps* em muitos outros. A metodologia é generalizada para lidar com o caso de múltiplas docas, com a finalidade de aproximar o problema de um sistema de *cross-docking* real, alcançando resultados que superam os atuais apresentados na literatura. É importante ressaltar que, o algoritmo híbrido proposto demonstra otimalidade em várias instâncias.

Uma abordagem de resequenciamento eficiente e um novo algoritmo para resolver

o problema de sequenciamento de caminhões com múltiplas docas sob incerteza no tempo de chegada dos caminhões é proposta e testada. Dois problemas de otimização diferentes são estudados e três metodologias comparadas a fim de comprovar a aplicabilidade da estratégia proposta. Os resultados demonstram que nossa metodologia pode apoiar os gerentes em suas operações diárias de *cross-docking*, uma vez que lidam com dados dinâmicos e incertos e resolvem problemas *online*.

A principal contribuição deste trabalho é o desenvolvimento de abordagens altamente eficientes que resolvem várias instâncias de tamanhos pequenos, médios e grandes, superando, até onde sabemos, todos os trabalhos publicados anteriormente.

A.6 Organização do Texto

Este trabalho é composto por seis capítulos. No capítulo 1 é introduzido o tema da pesquisa e apresentado os objetivos e contribuições do trabalho. O capítulo 2 apresenta um estudo teórico de trabalhos relacionados ao tema *cross-docking* e as principais publicações que lidam com sequenciamento de caminhões em um centro de *cross-docking*. O capítulo 3 trata o problema de sequenciamento com duas docas, no qual apresentamos o modelo matemático, a relaxação Lagrangiana, a metaheurística híbrida Lagrangiana proposta, resultados e conclusões. O problema de sequenciamento é generalizado no capítulo 4 para o caso de múltiplas docas, onde apresentamos a formulação proposta, relaxação Lagrangiana, algoritmo de resolução, assim como os resultados alcançados e as conclusões. No capítulo 5 uma abordagem de resequenciamento para o problema de sequenciamento de caminhões com múltiplas docas sob incerteza no tempo de chegada dos caminhões é detalhada e um novo algoritmo de resequenciamento eficiente é proposto e testado. Por fim, as conclusões dos trabalhos desenvolvidos são apresentadas no capítulo 6.

A.7 Publicações

Durante o primeiro ano do doutorado foi apresentado o trabalho completo intitulado: O problema de sequenciamento de caminhões em um centro de *cross-docking* com duas máquinas, no XLIX SBPO - Simpósio Brasileiro de Pesquisa Operacional, realizado na FURB - Fundação Universidade Regional de Blumenau, em Blumenau, SC, de 27 a 30 de agosto de 2017.

Em 2018, foi submetido o artigo intitulado '*A hybrid Lagrangian metaheuristic for the cross-docking flow shop scheduling problem*', na revista *European Journal of*

Operational Research (revista a qual o qualis é A1 para engenharias III), tendo sido este artigo aceito e publicado em novembro de 2018 (Apêndice D.1).

Submetemos um segundo artigo que trata o problema de sequenciamento considerando incerteza no tempo de chegada dos caminhões em um centro de *cross-docking* com múltiplas docas. Em contraste com as estratégias de sequenciamento existentes, uma nova abordagem de resequenciamento é proposta, testes computacionais foram realizados para comprovar a aplicabilidade em ambientes dinâmicos e com incertezas.

Appendix B

Considerations about WSPT-TRD

Considering the problem $1||C_{max} + \sum I_j W_j$, the algorithm WSPT-TRD obtains an optimal solution.
The objective function has two criteria, C_{max} and $\sum I_j W_j$. Regardless of the situation, C_{max}, will always aim to allocate the jobs as soon as possible. However, the $\sum I_j W_j$ depend on the weights values. Therefore, the proof is divided into two parts.

Part I: If $w_j^2 \geq 0$ (see Figure B.1).
In this case, the two criteria of the objective function are not in conflict. The criteria C_{max} and $\sum I_j W_j$ aims to schedule the jobs as soon as possible. As there is no idle time between jobs, for C_{max} any sequence starting at beginning of time horizon (t_i) is optimal. In this way, the only existing criteria becomes $\sum I_j W_j$ and its optimal WSPT rule is optimal for the problem.

Part II: If $w_j^2 < 0$.
In this case, the criteria are in conflict. C_{max} aim to schedule the jobs at beginning (t_i) of the time horizon, while $\sum I_j W_j$ at the end (t_f). Consider an optimal sequence generated by WSPT rule with one subset S' of jobs with $-1 < \sum w_j^2 < 0$. Suppose, by contradiction, that this subset must be allocated at the end (t_f) of the time horizon. Initially the sequence is all allocated at the beginning (t_i) of the time (see Figure B.2a). When the schedule S' with $-1 < \sum w_j^2 < 0$ is moved in one unit of the time horizon, its C_{max} increase one unit, but $\sum I_j W_j$ decrease $\sum w_j^2$. As C_{max} - $\sum w_j^2 > 0$ the objective function increase and the optimality is contradicted. Therefore, the $\sum w_j^2$ must be less or equal to 1 for allocate the jobs at the end (see Figure B.2b). It should be noted that for the case $\sum w_j^2 = -1$, these jobs are indifferent. □

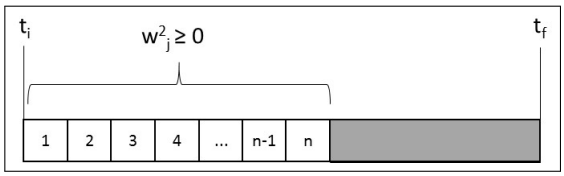
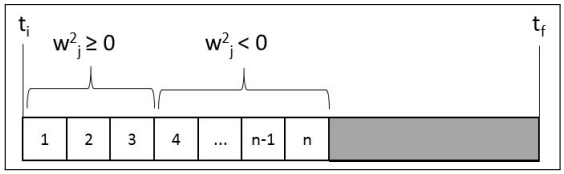
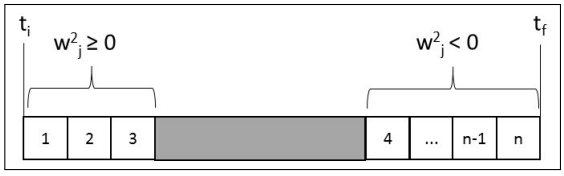


Figure B.1: Sequence with positive weights



(a) Initial Schedule



(b) Optimal Schedule

Figure B.2: Sequence with negative and positive weights

Appendix C

Sequential Parameter Optimization Toolbox (SPOT)

Due to its high efficiency, the Sequential Parameter Optimization Toolbox (SPOT) is the defined method to determine the parameters vector of Volume Algorithm. A parameters vector comprises the values of different parameters of the Volume Algorithm, i.e. it represents a candidate solution of a parameter setting. SPOT is an implementation of the Sequential Parameter Optimization (SPO), which is an iterative model-based method of tuning algorithms. The tuning process is based on the parameters' data, and their utility delivered by the performance of the volume algorithm. SPO performs a multi-stage procedure where the model is updated at each iteration with a set of new vectors and new predictions of utility in order to improve the algorithm's efficiency.

The goal of SPOT is the determination of good parameters settings for heuristic algorithms. It provides statistical tools for analyzing and understanding algorithm's performance. SPOT is implemented as a R package, and is available in the R archive network at `http://cran.r-project.org/web/packages/SPOT/index.html`. Further explanation about SPOT can be obtained from Bartz-Beielstein (2010), which provides an exemplification on how SPOT can be used for automatic and iterative tuning. Bartz-Beielstein and Zaefferer (2012) also give an introductory overview about tuning with SPOT.

The key elements of the SPOT methodology are algorithm design (D_a) and problem design (D_p). The first one defines ranges of parameter values that influence the behavior of an algorithm, such as crossover rate. These parameters are treated as variables $p_a \in D_a$ in the tuning algorithm, where p_a represents the vector of parameters settings. D_p refers to variables related to the tuning optimization problem, e.g. the search space dimension.

SPOT is composed by two phases: the build of the model and its sequential im-

provement. Phase 1 determinates an initial designs population from the algorithm's parameter space. The observed algorithm is run k times for each design, where k is the number of repetitions performed for each parameter setting and is increased in each run. Phase 2 leads to the efficiency of the approach and is characterized by the following loop:

- update the model given the obtained data;
- generate design points and predict their utility by sampling the model;
- choose the best design vectors and run $k+1$ times the algorithm for each of them;
- new design points are added to the population and the loop restarts if the termination criteria is not reached.

The loop simulates the use of different parameters settings, it is subject to interactions between parameters and random effects into the experiment. SPOT uses results from algorithm runs to build up a meta model to tune algorithms in a reproducible way.

C.0.1 Experimental setup

Classical works about SPOT define three different layers to analyze parameter tuning. The first one is the *application layer*, considered in our case as the two-machine flow shop problem with cross-docking constraints. The objective function and the problem parameters are defined at this layer. The *algorithm layer*, is related to the representation of the heuristic algorithm and its required parameters are the ones that determine the algorithm's performance. And finally, the *design layer*, where is the tuning method, tries to find good parameter settings for the algorithm layer.

In this way we face two optimization problems: problem solving and parameter tuning. The problem solving covers the application layer and the Volume Algorithm of the algorithm layer and aims to find an optimal solution for the problem. The parameters tuning uses a tuning method to find the best parameter values for the Volume Algorithm based on lower bounds found. The fitness value is the quality measure of the first optimization, which depends on the problem instance to be solved. Utility is the quality measure for parameter tuning, which reflects the performance of the Volume Algorithm for a vector of parameters.

Thereby, the experimental setup consists of an application layer represented by twenty instances of the two-machine flow shop problem, showed on Table C.1. To define the region of interest (ROI) of the tuning algorithm, the type and the lower and upper bounds of the volume algorithm's parameters are summarized on Table C.2.

Table C.1: Sampling of instances

Type	Instances
Y1	5-3-4-1
Y2	5-4-4-1
Y3	5-5-4-1
Y4	5-6-4-1
Y5	5-7-4-1
Y6	10-6-9-1
Y7	10-8-9-1
Y8	10-10-9-1
Y9	10-12-9-1
Y10	10-14-9-1
Y11	5-3-4-2
Y12	5-4-4-2
Y13	5-5-4-2
Y14	5-6-4-2
Y15	5-7-4-2
Y16	10-6-9-2
Y17	10-8-9-2
Y18	10-10-9-2
Y19	10-12-9-2
Y20	10-14-9-2

Table C.2: Parameter bounds for tuning the Volume Algorithm

Parameter	Lower bound	Upper bound	Type
π	0.0005	0.0100	FLOAT
MaxWaste	5	30	INT
factor	0.1	1.0	FLOAT
α_{max}	0.20	0.95	FLOAT
st	0.6	1.8	FLOAT
α	0.05	0.90	FLOAT
yellow	0.20	0.95	FLOAT
green	1.0	2.0	FLOAT

C.0.2 Results of the experiment

The SPOT algorithm, implemented in R package, is connected with the volume algorithm implemented on C++ with the aid of one callString, as presented below. The computer used in the tests is an Intel (R) Xeon (R) CPU X5690 @ 3.47GHz with 24 processors, 132 GB of RAM, and Ubuntu Linux operating system.

$callString \leftarrow paste(\text{“./HgR ”}, \pi, MaxWaste, factor, \alpha_{max}, st, \alpha, yellow, green)$
$call \longleftarrow system(callString, intern = TRUE)$
$\#read\ the\ results$
$y = read.table(\text{“Results.txt”})$

The Table C.3 presents the four best results found by SPOT method for the volume algorithm. The first column indicates the iteration of SPOT, the second one presents the indicated parameters vector and the third column indicate the solution value obtained by the tested instances (lower bounds).

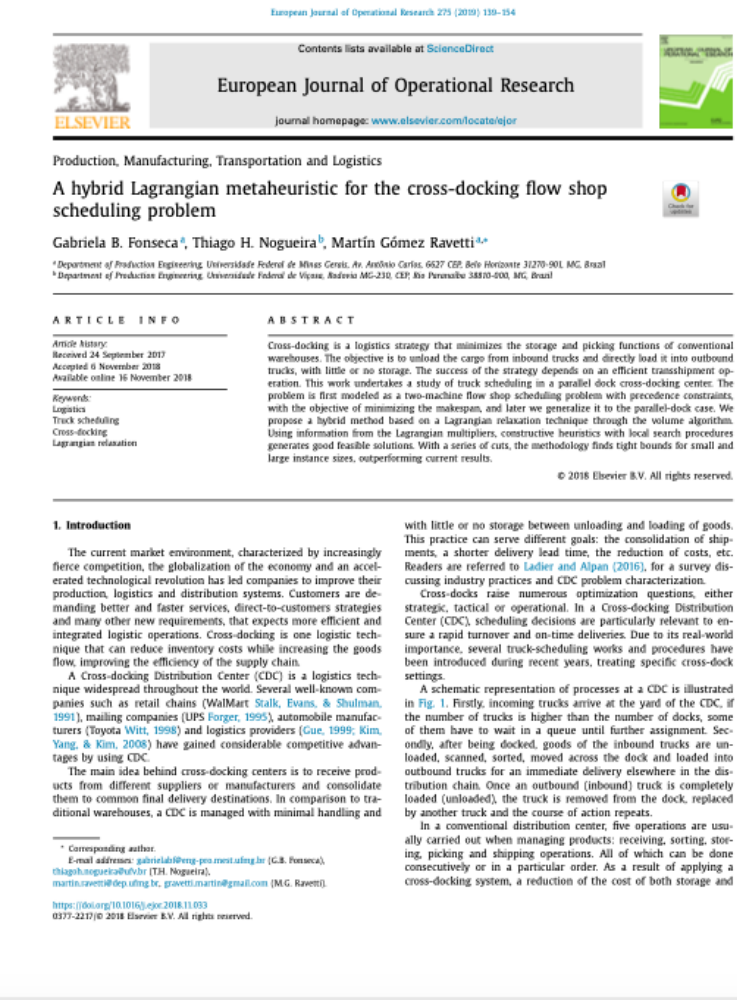
The best parameters vector indicated for the volume algorithm, that generated the best lower bounds is π=0.00679, MaxWaste=24, factor=0.87753, α_{max}=0.30337, st=1.41179, α=0.08300, yellow=0.48159 and green=1.56487.

Table C.3: Four best parameters settings for the volume algorithm

Iteration	Parameter								Solution value of the instances																
	π	MaxWaste	factor	α_{max}	st	α	yellow	green	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9	Y10	Y11	Y12	Y13	Y14	Y15	Y16	Y17	Y18	Y19	Y20
501	0.00708	28	0.22871	0.33265	1.01875	0.21709	0.80291	1.13994	24	14	30	20	30	36	46	56	60	66	155	137	191	328	414	337	537	558	629	793
502	0.00527	9	0.34752	0.75030	1.71397	0.73405	0.58396	1.56980	24	15	29	21	30	37	46	56	60	66	157	154	191	341	414	337	563	558	629	793
503	0.00688	20	0.62855	0.83046	1.17734	0.28130	0.60572	1.40758	25	15	30	21	30	37	46	56	60	66	118	136	191	307	414	337	531	558	629	793
504	0.00679	24	0.87753	0.30337	1.41179	0.08300	0.48159	1.56487	26	15	30	20	30	32	46	56	60	66	95	136	191	307	414	337	531	558	629	793

Appendix D

Publication



European Journal of Operational Research 275 (2019) 139–154

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

A hybrid Lagrangian metaheuristic for the cross-docking flow shop scheduling problem

Gabriela B. Fonseca[a], Thiago H. Nogueira[b], Martín Gómez Ravetti[a,*]

[a] Department of Production Engineering, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627 CEP, Belo Horizonte 31270-901, MG, Brazil
[b] Department of Production Engineering, Universidade Federal de Viçosa, Rodovia MG-230, CEP, Rio Paranaíba 38810-000, MG, Brazil

ARTICLE INFO

Article history:
Received 24 September 2017
Accepted 6 November 2018
Available online 16 November 2018

Keywords:
Logistics
Truck scheduling
Cross-docking
Lagrangian relaxation

ABSTRACT

Cross-docking is a logistics strategy that minimizes the storage and picking functions of conventional warehouses. The objective is to unload the cargo from inbound trucks and directly load it into outbound trucks, with little or no storage. The success of the strategy depends on an efficient transshipment operation. This work undertakes a study of truck scheduling in a parallel dock cross-docking center. The problem is first modeled as a two-machine flow shop scheduling problem with precedence constraints, with the objective of minimizing the makespan, and later we generalize it to the parallel-dock case. We propose a hybrid method based on a Lagrangian relaxation technique through the volume algorithm. Using information from the Lagrangian multipliers, constructive heuristics with local search procedures generates good feasible solutions. With a series of cuts, the methodology finds tight bounds for small and large instance sizes, outperforming current results.

1. Introduction

The current market environment, characterized by increasingly fierce competition, the globalization of the economy and an accelerated technological revolution has led companies to improve their production, logistics and distribution systems. Customers are demanding better and faster services, direct-to-customers strategies and many other new requirements, that expects more efficient and integrated logistic operations. Cross-docking is one logistic technique that can reduce inventory costs while increasing the goods flow, improving the efficiency of the supply chain.

A Cross-docking Distribution Center (CDC) is a logistics technique widespread throughout the world. Several well-known companies such as retail chains (WalMart Stalk, Evans, & Shulman, 1991), mailing companies (UPS Forger, 1995), automobile manufacturers (Toyota Witt, 1998) and logistics providers (Gue, 1999; Kim, Yang, & Kim, 2008) have gained considerable competitive advantages by using CDC.

The main idea behind cross-docking centers is to receive products from different suppliers or manufacturers and consolidate them to common final delivery destinations. In comparison to traditional warehouses, a CDC is managed with minimal handling and with little or no storage between unloading and loading of goods. This practice can serve different goals: the consolidation of shipments, a shorter delivery lead time, the reduction of costs, etc. Readers are referred to Ladier and Alpan (2016), for a survey discussing industry practices and CDC problem characterization.

Cross-docks raise numerous optimization questions, either strategic, tactical or operational. In a Cross-docking Distribution Center (CDC), scheduling decisions are particularly relevant to ensure a rapid turnover and on-time deliveries. Due to its real-world importance, several truck-scheduling works and procedures have been introduced during recent years, treating specific cross-dock settings.

A schematic representation of processes at a CDC is illustrated in Fig. 1. Firstly, incoming trucks arrive at the yard of the CDC, if the number of trucks is higher than the number of docks, some of them have to wait in a queue until further assignment. Secondly, after being docked, goods of the inbound trucks are unloaded, scanned, sorted, moved across the dock and loaded into outbound trucks for an immediate delivery elsewhere in the distribution chain. Once an outbound (inbound) truck is completely loaded (unloaded), the truck is removed from the dock, replaced by another truck and the course of action repeats.

In a conventional distribution center, five operations are usually carried out when managing products: receiving, sorting, storing, picking and shipping operations. All of which can be done consecutively or in a particular order. As a result of applying a cross-docking system, a reduction of the cost of both storage and

* Corresponding author.
E-mail addresses: gabrielabf@eng-pro.mest.ufmg.br (G.B. Fonseca), thiagoh.nogueira@ufv.br (T.H. Nogueira), martin.ravetti@dep.ufmg.br, gravetti.martin@gmail.com (M.G. Ravetti).

https://doi.org/10.1016/j.ejor.2018.11.033
0377-2217/

Figure D.1: Article published in European Journal of Operational Research.

Appendix E

Complementary analysis of subsection results 5.5.3

Figures E.1 and E.2 summarizes the results in terms of average tardiness objective function value. On the left are the results for the C_{max} problem. On the right, the WC problem. As previously mentioned, we can see that the RA presents the lowest values in all cases for C_{max} problem. For WC problem, RA presents excellent results for environments with high and low levels of uncertainties.

Figures E.3 and E.4 reinforce the results indicated in Table 5.2. The graphs on the left (C_{max} problem) show that the average number of trucks delayed is lower for all instances groups in RA. On the right, the WC problem graphs show that the average number of trucks delayed in RA is also very low. The results presented above demonstrate the benefits of our rescheduling algorithm. The smaller the number of trucks delayed, the greater the number of customers who had their deliveries made on the promised date, increasing customer satisfaction and consequently the level of logistical service.

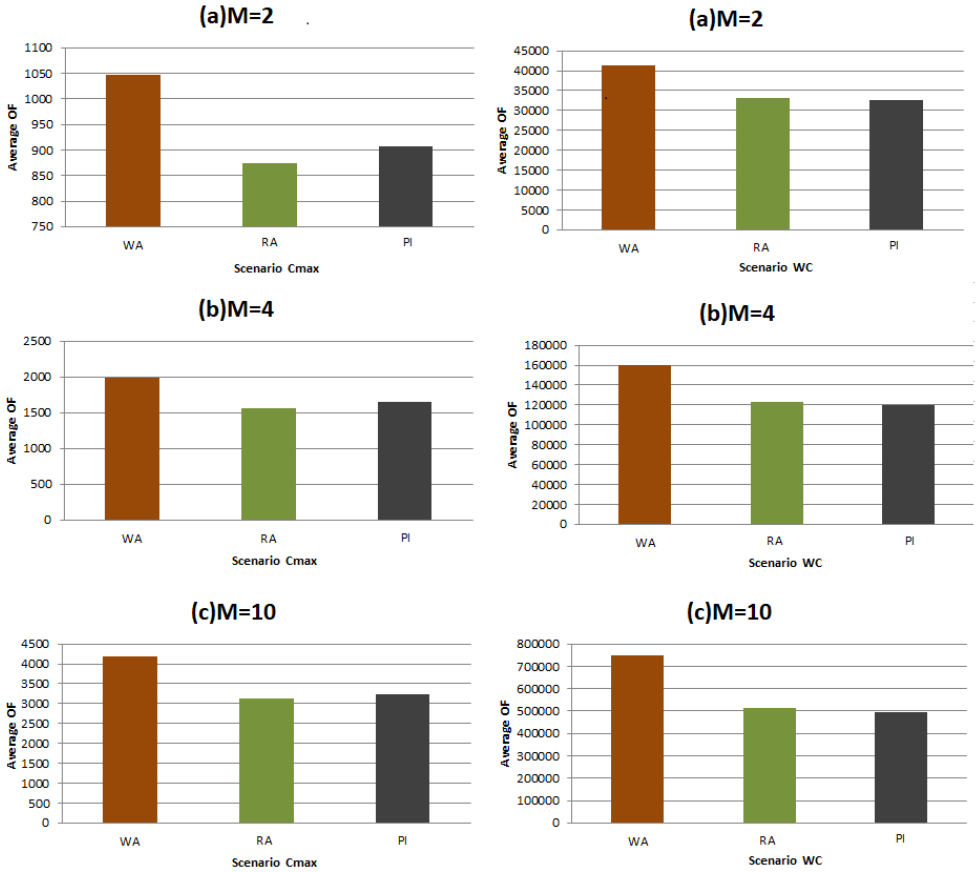


Figure E.1: Average objective function value of each instance group considering the same number of machines at each stage: (a) $M = 2$, (b)$M = 4$, (c) $M = 10$.

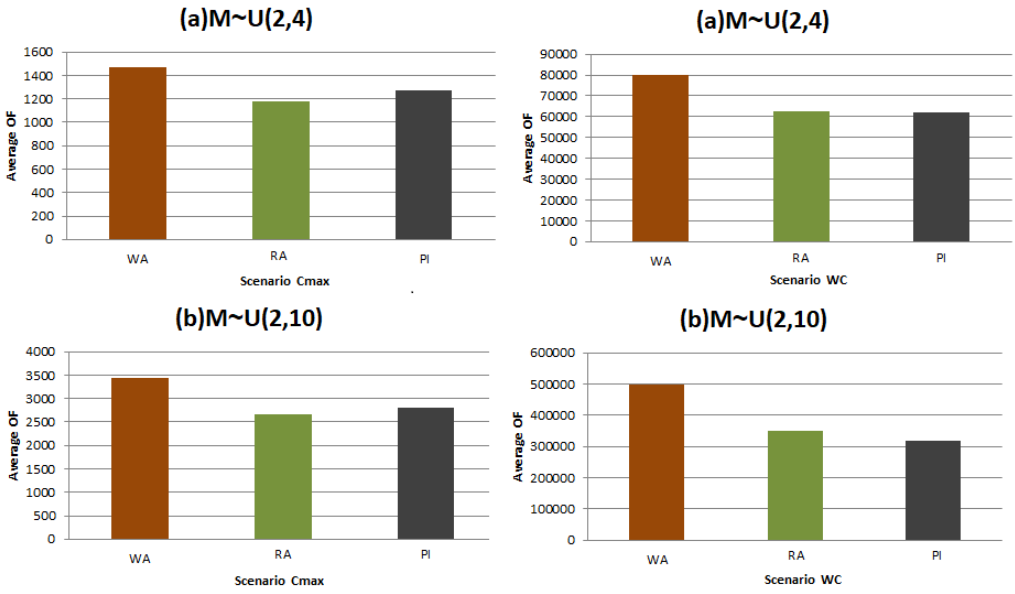


Figure E.2: Average objective function value of each instance group considering a random number of machines at each stage: (a) $M \sim U(2, 4)$, (b) $M \sim U(2, 10)$.

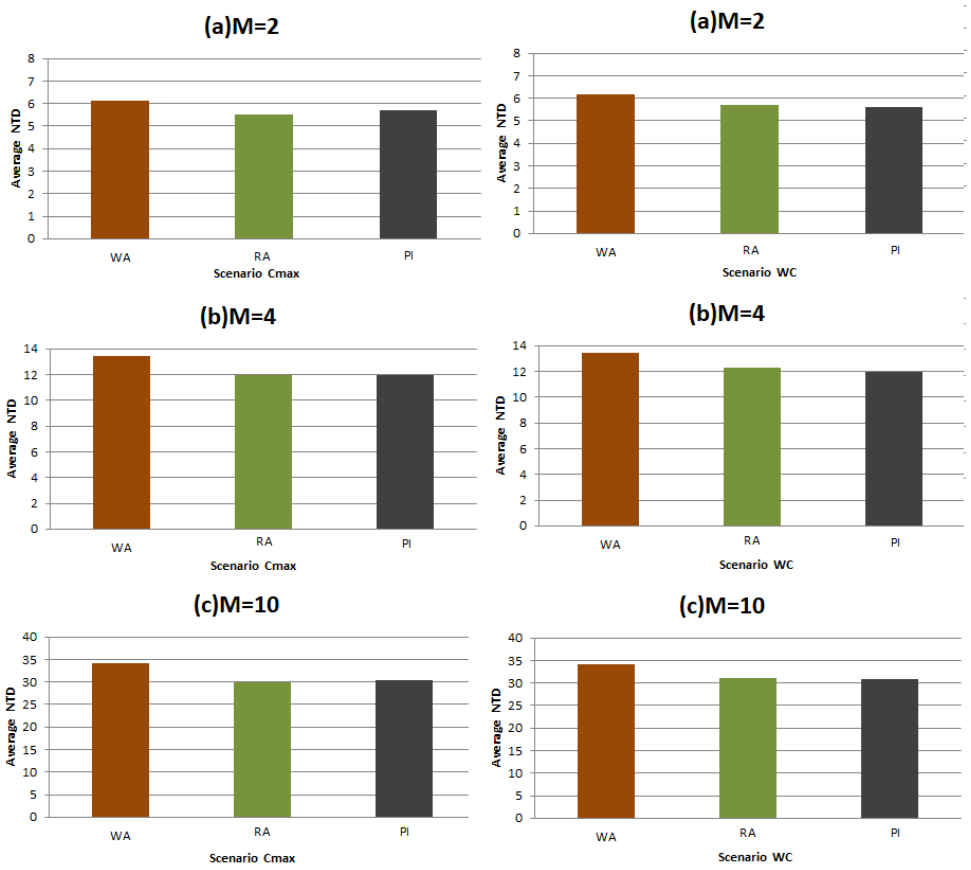


Figure E.3: Average number of trucks delayed of each instance group considering the same number of machines at each stage: (a) $M = 2$, (b)$M = 4$, (c) $M = 10$.

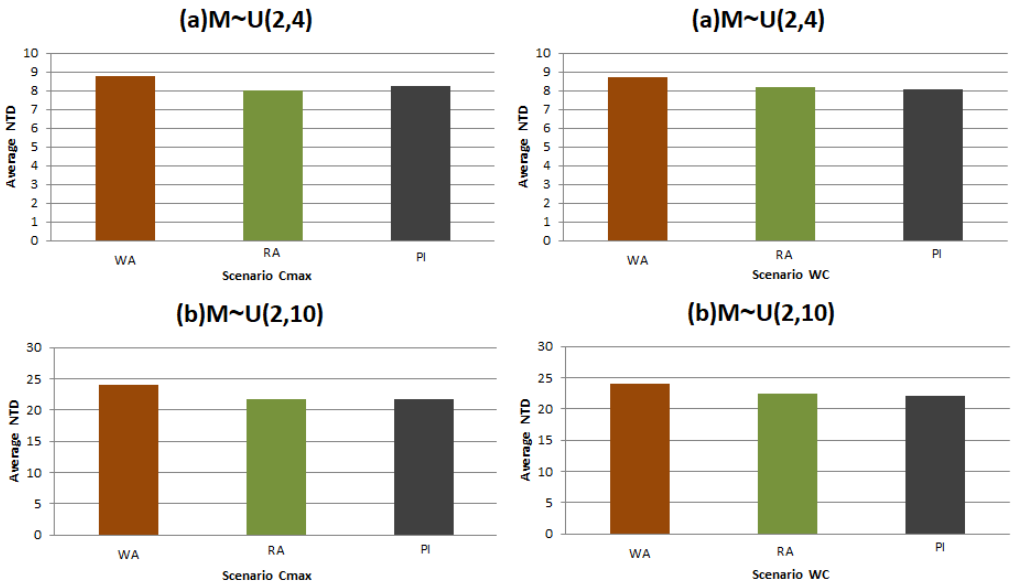


Figure E.4: Average number of trucks delayed of each instance group considering a random number of machines at each stage: (a) $M \sim U(2, 4)$, (b) $M \sim U(2, 10)$.

Appendix F

Comparative analysis of the parameters of the multi-dock truck scheduling problem under arrival time uncertainty

In this appendix, we present a comparative analysis of the parameters of the multi-dock truck scheduling problem under arrival time uncertainty. We present the results for C_{max} and WC problems, for the five groups of instances ($m_i = \{2, 4, 10, U(2, 4), U(2, 10)\}$). Five metrics are discussed to evaluate the impact of different metrics on the average number of trucks delayed in the three methodologies studied. The study on different metrics comes from the fact that the complexity of the operation usually challenges the manager, and different perspectives on the quality of the solution can shed light on the situation. The first metric considers the number of outbound trucks per the number of inbound trucks (n_2/n_1); the second metric considers the number of inbound trucks per outbound trucks (n_1/n_2); the third metric is based on the average r_d. The fourth is the average *Real* r_d, as defined in the subsection 5.5.2. Finally, the last metric is based on the standard deviation of the release dates for the inbound trucks.

The analysis considers the same number of machines at each stage (case in which $m_i = 2$), as an example, are graphed in Figures F.1 and F.2. In the first analysis (n_2/n_1), when the number of outbound trucks is greater than inbound trucks in a CDC, the average number of trucks delayed grows. As expected, the greater the number of outbound trucks, the more significant difficulties the methods encounter in maintaining deliveries on the promised dates, especially for the WA methodology. However, when the number of inbound trucks per outbound trucks (n_1/n_2) is considered, we observed that the average number of trucks delayed tends to decrease as the fraction increases.

The decrease in the number of outbound trucks makes delivery within their due dates more possible.

By analyzing the metrics average r_d and average *Real* r_d, it is possible to see that the average number of trucks delayed increases drastically over the first hours of CDC operation (r_d ranging from 480 to 720), afterward, it tends to stabilize. This fact occurs probably since truck arrivals can concentrate between 8 am and 12 pm, causing further delays in this period. The last analysis shows that there is more flexibility in scheduling the trucks when the standard deviation is null. On the other hand, as the standard deviation increases, the difficulty of scheduling trucks increases, generating a more significant impact on the number of delayed trucks.

We further observe that regardless of the metric evaluated, the proposed algorithm RA has very similar behavior to PI, proving the advantage of using a rescheduling methodology. Another interesting point is that the behavior of the data in the graphs, for both the C_{max} and WC problems, are very similar, which indicates that the metrics seem to have the same impact on both problems.

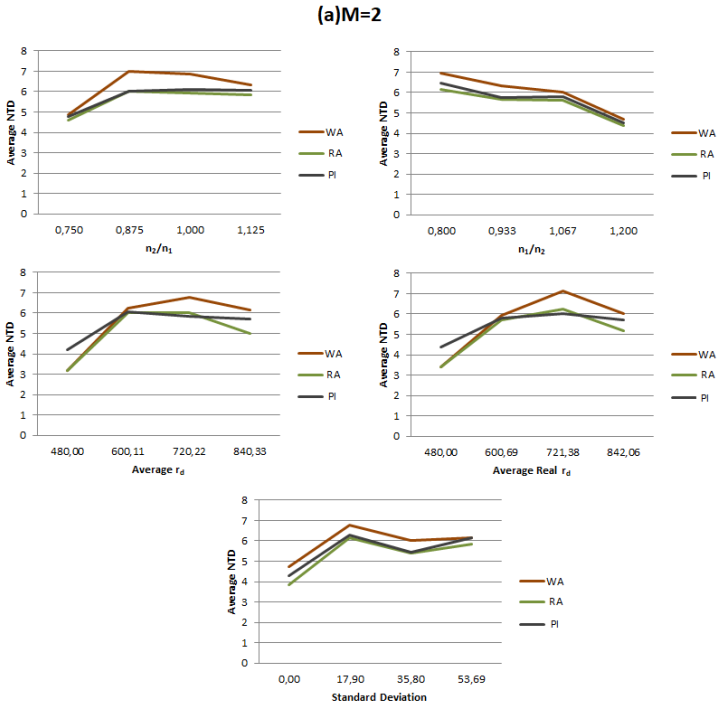


Figure F.1: Average number of trucks delayed for different metrics to the C_{max} problem, considering the same number of machines at each stage: (a) $M = 2$.

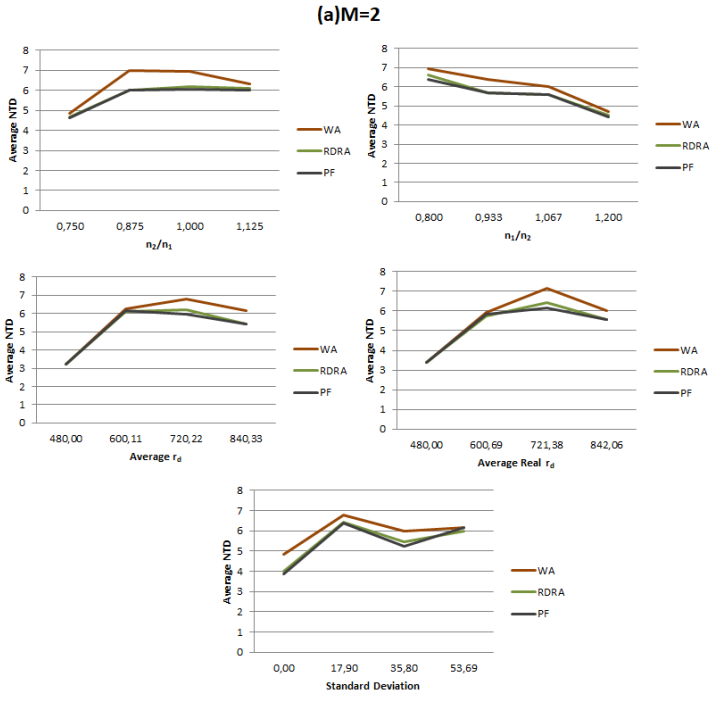


Figure F.2: Average number of trucks delayed for different metrics to the WC problem, considering the same number of machines at each stage: (a) $M = 2$.