Universidade Federal de Minas Gerais

Programa de Pós-Graduação em Engenharia de Produção

Departamento de Engenharia de Produção / Escola de Engenharia

Problema de Sequenciamento de Caminhões em Centros de *Crossdocking* com múltiplas docas

Priscila Mara Cota

Orientador: Prof. Dr. Martín Gomez Ravetti

Belo Horizonte
Maio, 2015

Universidade Federal de Minas Gerais

Programa de Pós-Graduação em Engenharia de Produção

Departamento de Engenharia de Produção / Escola de Engenharia

Problema de Sequenciamento de Caminhões em Centros de *Crossdocking* com múltiplas docas

Priscila Mara Cota

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Minas Gerais como requisito para a obtenção do título de Mestre em Engenharia de Produção.

Área de Concentração: Produção e Logística

Linha de Pesquisa: Modelos e Algoritmos de Produção e de Redes

Orientador: Prof. Dr. Martín Gomez Ravetti

Belo Horizonte
Maio, 2015

À minha família;

Resumo

Esta dissertação aborda o sequenciamento de caminhões em um Centro de *Cross-docking* com múltiplas docas de processamento. O problema é formulado como um sequenciamento do tipo *flowshop* híbrido de dois estágios, sujeito à restrições de *crossdocking*, cujo objetivo é minimizar o *makespan*. Essas restrições proíbem os *jobs* do segundo estágio de iniciar seu processamento antes da conclusão de seus *jobs* precedentes. Um modelo de programação linear inteira mista indexado no tempo é proposto com a finalidade de resolver o problema para instâncias de pequeno porte, através de um solver comercial. Para resolver médias e grandes instâncias quatro heurísticas construtivas são propostas e analisadas. Experimentos computacionais foram realizados de forma a comparar os resultados obtidos aos apresentados na literatura.

Palavras-Chave: Sequenciamento de caminhões, Programação linear inteira mista indexado no tempo, heurísticas construtivas.

Abstract

This dissertation undertakes the study of truck scheduling in a crossdocking station, with multiple docks in parallel. The problem is formulated as a two-stage hybrid flowshop problem, minimizing the makespan, and subject to crossdocking constraints. These constraints forbid a job in the second stage to be processed until the conclusion of its precedent jobs. We proposed a time-indexed mixed integer linear programming model able to solve small instances through a commercial solver. For moderate and large size instances, four constructive heuristics are proposed and tested. Computational experiments were performed to compare these results with those reported in the literature.

Palavras-Chave: Trucks scheduling, Time-indexed mixed integer linear programming, constructive heuristics.

Agradecimentos

Primeiramente à *Deus*, que está sempre no comando de toda minha vida.

Ao professor Doutor Martín Gomez Ravetti pela orientação, que guia esse trabalho com infinita competência.

Ao professor Doutor Thiago Henrique Nogueira, por sua grande contribuição para a qualidade dessa dissertação.

À minha família.

Aos professores, funcionários e colegas do Departamento de Engenharia de Produção.

À FAPEMIG pelo incentivo financeiro concedido, que é imprescindível para a dedicação integral ao mestrado.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O presente trabalho trata do problema de sequenciamento de caminhões em um Centro de *Crossdocking* (CCD). *Crossdocking* é uma estratégia logística utilizada atualmente por muitas empresas pertencentes a diferentes setores industriais. Onde o objetivo é minimizar o tempo de processamento de cargas dentro desses centros, desde o momento em que elas são descarregadas dos veículos de entrada até o momento em que o último veículo de saída é expedido rumo aos clientes.

O problema é modelado de forma análoga a um problema de *flowshop* híbrido de dois estágios com máquinas idênticas e paralelas e com restrições que determinam um conjunto de *jobs* precedentes para os *jobs* do segundo estágio, chamadas de restrições de *crossdocking*. Esse modelo foi proposto por Chen e Song [9]. Assim, as máquinas paralelas são análogas às docas de entrada e saída dos caminhões no CCD e os *jobs* são associados às cargas dos caminhões. Os tempos de processamento dos *jobs* no primeiro e no segundo estágio respectivamente correspondem às atividades de descarregamento e carregamento dos caminhões.

Este projeto de mestrado trata sobre formas eficientes para resolver o problema de sequenciamento de caminhões em um centro de *crossdocking* com múltiplas docas que minimize o tempo total de conclusão das tarefas. Para isso, definimos que a abordagem aqui proposta será a mesma dos autores supracitados com o intuito de ampliar e melhorar os resultados presentes na literatura. Assim, propomos primeiramente um modelo de programação linear inteira mista indexado no tempo, que resolve através de um solver comercial instâncias de pequena escala, e posteriormente quatro heurísticas construtivas polinomiais com a finalidade de resolver médias e grandes instâncias.

1.1 Objetivos

O objetivo principal deste trabalho é melhorar os resultados existentes na literatura a respeito do problema de sequenciamento de caminhões em centros de *crossdocking* com múltiplas docas de processamento de forma a minimizar o tempo total de conclusão das tarefas. Para isso, será proposto um modelo de programação inteira com formulação baseada em indexação do tempo, que, em geral, apresenta bons limites inferiores. Esses limites serão analisados através da relaxação linear e comparados aos limites produzidos pelo modelo presente na literatura. Posteriormente quatro heurísticas construtivas são estudadas com a finalidade de resolver o problema de uma maneira mais realística, tratando de instâncias maiores e que não são resolvidas pelo modelo exato devido a complexidade do problema.

1.2 Justificativa

Este trabalho se justifica por dois fatores, um de cunho acadêmico e outro de cunho prático. Do ponto de vista prático devido ao impacto positivo que a abordagem de centros de *crossdocking* podem trazer para um sistema logístico, como abordado na Seção 2.1, tratado da melhor maneira, a fim de se adaptar a casos reais de *crossdocking*. Academicamente, se justifica por apresentar um novo modelo matemático, bem como de novas heurísticas construtivas polinomiais, de forma a ampliar e melhorar os resultados presentes na literatura até o momento.

1.3 Organização do Texto

O texto é organizado da seguinte maneira: O Capítulo 2 aborda a Contextualização deste trabalho, definindo o que são Centros de *Crossdocking*, revisando bibliograficamente o tema e definindo de maneira formal o problema. No Capítulo 3 as modelagens matemáticas para o problema são tratadas e o modelo matemático indexado no tempo proposto é apresentado, bem como, suas considerações e definições. No Capítulo 4, todos os resultados à respeito do modelo são explicitados, primeiramente o modelo é comparado ao de Chen e Song [9] por meio da relaxação linear, posteriormente prova-se a dominância do modelo proposto, uma vez definida a dominância, o modelo é testado de uma maneira mais detalhada. O Capítulo 5 apresenta as heurísticas polinomiais. No Capítulo 6, os resultados das heurísticas são analisados, a partir de uma comparação de desempenhos. Finalmente, o Capítulo 7 conclui a dissertação.

Capítulo 2

Contextualização

Inicialmente, definiremos o que são Centros de *Crossdocking* (CCD), evidenciando vantagens de seu uso e sua aplicabilidade na prática. Em seguida, será realizado o exame da literatura com diversos enfoques e níveis estratégicos em relação aos problemas asssociados aos CCD estudados. O foco será em publicações que tratem do problema de sequenciamento de caminhões em CCD. Finalmente, o problema estudado será formalmente definido.

2.1 Centros de *Crossdocking*

Em redes de distribuição tradicionais, existem vários fornecedores de produtos distintos que realizam o transporte de cargas até vários pontos dessa rede, com a finalidade de atender aos clientes de uma maneira geral (galpões, atacadistas, varejistas). Dessa forma existem diversos projetos para a cadeia de suprimento. Muitas vezes, as redes de distribuição de mercadorias são ineficientes e geram gastos desnecessários por possuirem estruturas baseadas em modelos históricos ou em experiências passadas.

Um modelo usual de rede logística é a configuração ponto a ponto ou encomenda direta. Nessa configuração cada fornecedor entrega suas mercadorias diretamente a todos os seus clientes e, cada cliente recebe suas encomendas de cada fornecedor, como apresentado na Figura 2.1.

Este *layout* possui suas vantagens: não existe custo operacional de armazéns e os tempos de entregas são reduzidos. Entretanto, ele pode ser altamente ineficiente quando vários clientes contemplam os mesmos fornecedores. Neste caso, caminhões menores devem ser utilizados para o transporte ou então, caminhões serão usados com carga reduzida. Em ambos os casos, o receptor deve possuir uma capacidade

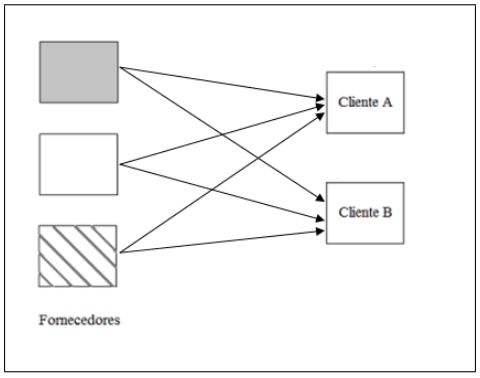

Figura 2.1: Modelo comum de distribuição.

de estocagem considerável. Além disso, o custo de transporte global é alto, pois muitas entregas devem ser feitas para atender todos os consumidores finais da rede, o que exige uma frota com grande número de veículos de transporte. Se um mesmo caminhão sai com destino a vários clientes, a partir da primeira entrega, o caminhão passa a trafegar com carga parcial. Essas complicações deram origem aos centros tradicionais de distribuição, que são unidades especializadas com a função de consolidar e armazenar produtos de diversas fontes para serem distribuídos a pontos de revenda ou clientes finais.

Em um centro tradicional de distribuição, cargas são recebidas e estocadas. Quando um cliente faz um pedido, os itens correspondentes ao pedido são coletados do estoque e despachados. Assim, há quatro principais funções no centro tradicional: receber, estocar, coletar (*picking*) e despachar itens. *Crossdocking* é uma abordagem que elimina ou reduz drasticamente as duas funções mais dispendiosas dos centros tradicionais de distribuição que são a estocagem e coleta dos produtos, para isso, um Centro de *Crossdocking* (CCD) funciona com um estoque limitado ou, se possível, nulo.

Os CCD não realizam as atividades de estocagem e coleta, pois a carga recebida por diversos fornecedores é imediatamente preparada para ser transferida para a área de despacho e destinada aos clientes. Os Centros de *Crossdocking* operam recebendo carretas de diversos pontos de fornecimento, cada veículo é recebido em uma doca específica. Dentro do centro, as cargas são retiradas, separadas, combinadas e recarregadas em carretas de saída, de acordo com os pedidos específicos dos clientes. Essas carretas então deixam a instalação com carga combinada, composta por produtos de diversos fornecedores, dedicada a um cliente ou mais clientes ou destino específico. A lógica dos CCD é ilustrada na Figura 2.2.

Segundo EAN *International* [1], a utilização do centro de crossdocking traz diversas vantagens para o sistema de distribuição, destacando-se:

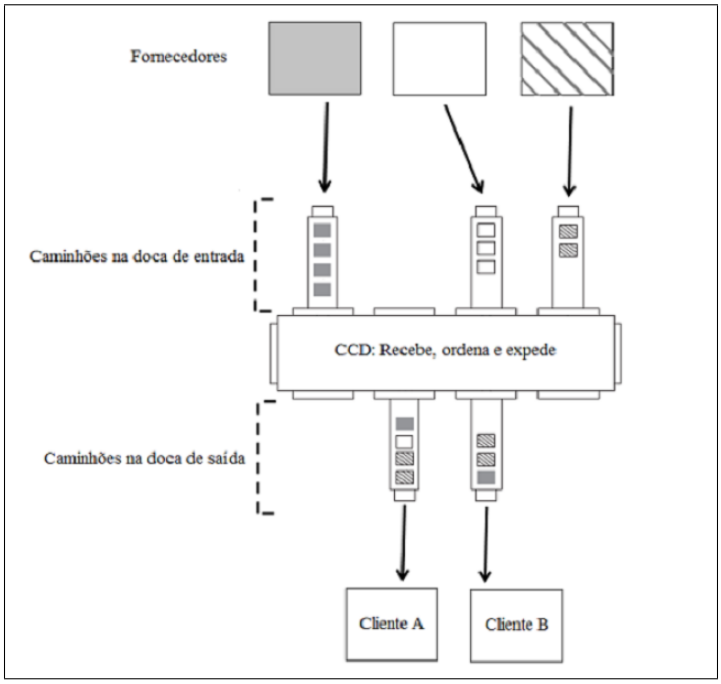

Figura 2.2: Rede de distribuição com *Crossdocking*. - Fonte: Adaptado de Lira [16]

1. Redução dos custos de estoque e eliminação dos problemas operacionais de armazenagem e coleta.
2. Redução do custo unitário de transporte, visto que o sistema opera com carga completa ou próxima da capacidade máxima nas carretas.
3. Redução da complexidade de entrega que é realizada de uma só vez com toda a variedade de produtos requerida pelo cliente.
4. Aumento da vida de prateleira do produto, uma vez que ele passará menos ou nenhum tempo em estoque intermediário.
5. Aumento da rotatividade do centro de *crossdocking*, já que o sistema opera com entregas em menor quantidade e mais frequentes.
6. Redução da falta de estoque nas lojas dos clientes, devido ao ressuprimento contínuo e mais frequente proporcionado pela rede.
7. Redução do estoque total em toda a cadeia de suprimentos, na qual o produto passa a fluir sem ser estocado.

Por outro lado, apesar das vantagens obtidas com essa nova estratégia logística, alguns cuidados devem ser tomados. Para que a cadeia de suprimentos funcione de maneira eficiente, todos os participantes da cadeia devem estar envolvidos na busca da excelência no funcionamento do sistema, tanto na gestão de materiais, quanto na gestão da informação. A utilização dessas informações permite que as

instalações planejem suas operações antes do recebimento das mercadorias, além de permitir também o planejamento e gerenciamento de suas capacidades. Ao adicionar um novo ponto de descarga e carga na cadeia, o processo de distribuição se torna mais lento, gerando também uma quantidade considerável de dupla manipulação de produtos. Dessa forma, é necessário que se implemente métodos eficientes de transferência de carga em um centro de *crossdocking*, nos quais as descargas e cargas são eficientemente sincronizadas de forma a manter o nível de estoque baixo e assegurar as entregas a tempo [6]. Essa necessidade deu origem aos problemas de sequenciamento em CCD, que é o escopo deste trabalho, cada vez mais importante diante da expansão dos CCD.

A estratégia de *crossdocking* nos últimos anos tem tido aplicações muito interessantes, e com resultados satisfatórios. Simchi-Levi et al. [20] nos mostra que empresas como Amazon.com, Coca-cola, Dell e Wal-Mart se tornaram referências em soluções inovadoras de gestão da cadeia de suprimentos. O Wal-Mart, por exemplo, se tornou o maior varejista mundial focando nas necessidades dos clientes de modo que o produto esteja onde e quando o consumidor desejar, com custos compatíveis. Para isto, foi fundamental a utilização de centros de *crossdocking*.

O problema será tratado nesta dissertação analogamente a um sequenciamento comum em linhas de produção de modo que docas recebem e descarregam ou carregam caminhões assim como máquinas processam ordens de produção (*jobs*). O problema pode ser considerado um problema de sequenciamento em ambiente de *flowshop* híbrido com dois estágios. Assim o problema consiste na determinação da sequência na qual caminhões são descarregados nas docas de entrada e carregados nas docas de saída, de forma a minimizar o tempo total de conclusão das tarefas(*makespan*).

2.2 Exame da Literatura

O aumento no uso de técnicas de *crossdocking* e a conseqüente maior exposição do tema tem motivado diversos autores a pesquisar novas maneiras de melhorar as operações associadas à prática. Existem diferentes abordagens a respeito dos CCD com enfoques nas diversas etapas do processo logístico. Boysen e Fliedner [6] e Belle et al. [4] focam na revisão dos trabalhos presentes na literatura que possuem o tema *crossdocking* como foco principal.

Centros de *crossdocking*, assim como unidades de produção, tratam problemas envolvendo a tomada de decisão, desde a concepção até o nível operacional. Boysen e Fliedner [6] definem os seguintes problemas enfrentados durante o ciclo

de vida de um CCD do nível estratégico ao operacional: localização do Centro de *Crossdocking*, *layout*, atribuição de docas, roteamento de veículos, sequenciamento de caminhões, sequenciamento interno de recursos e empacotamento e desempacotamento de cargas. O presente trabalho possui foco principal em dois desses problemas a atribuição de docas e o sequenciamento dos caminhões tanto de entrada quanto de saída.

Belle et al. [4] destacam que na literatura, às vezes, o número de docas é limitado a apenas uma ou duas. Nesses casos, o foco não é modelar um sistema real de *crossdocking*, mas obter informações importantes estudando modelos simplificados. Segundo os autores boas alocações de docas são muito importantes, uma vez que, podem aumentar a produtividade dos CCD e reduzir os custos de manuseio. O problema de alocação de docas procura definir a alocação ótima dos caminhões às docas que serão carregados e descarregados, de forma a minimizar o *makespan*, ou seja, o tempo necessário desde o descarregamento dos caminhões nas docas de entrada até o carregamento do último caminhão a ser despachado. O ideal em um CCD é que o número de docas seja igual ao número de caminhões, assim cada caminhão será alocado a uma doca e aspectos envolvendo tempo não serão levados em consideração. Entretanto, na maior parte das vezes isso não é possível e as docas passam a ser um recurso escasso e devem ser sequenciadas ao longo do tempo, dando origem ao problema de sequenciamento de caminhões.

A alocação de docas pode ser executada em um horizonte de curto a médio prazo [6]. Belle et al. [4] afirmam que muitos estudos na literatura tratam o problema a médio prazo. Onde cada doca é atribuída exclusivamente a destinos ou origens específicos por um certo período de tempo(em média 6 meses). Ainda segundo os autores docas fixas são um ponto positivo para os colaboradores que trabalham no CCD, devido a facilidade em conhecer as docas as quais devem encaminhar os caminhões, entretando essa prática pode apresentar um alto custo devido à redução da flexibilidade do centro. Por outro lado quando informações sobre a chegada de caminhões são fornecidas antecipadamente, a alocação de docas pode ser feita em um horizonte de curto prazo.

Tsui e Chang [21, 22] foram os primeiros a estudar o problema de atribuição de docas a médio prazo. Em [21] os autores apresentaram um problema genérico de atribuição de docas, no qual as docas de entrada e saída encontram-se em lados opostos de um CCD retangular. Dependendo da distância entre a doca de recebimento e da doca de expedição, o tempo gasto para se finalizar a tarefa de transbordo pode variar significativamente, assim como o número de equipamentos necessário para fazê-lo. Tem-se, então, que o objetivo de minimizar a distância entre docas resulta na minimização do esforço de se transportar a carga dentro do armazém.

O problema foi resolvido utilizando um algoritmo bilinear que retorna uma solução eficaz. Esse algoritmo gera inicialmente uma atribuição de portas de recebimento qualquer, e a partir dessa escolha, uma combinação de portas de despacho é selecionada para esta configuração de docas. Em seguida, encontra-se uma solução de portas de recebimento que consegue otimizar a configuração de portas de despacho encontrada no passo anterior. O algoritmo se repete enquanto houver melhora na solução. O ponto negativo desse algoritmo pode ser percebido facilmente, uma vez que, o tempo e a solução final dependem fortemente da solução inicial. Posteriormente, Tsui e Chang [22] propuseram através de um algoritmo *Branch and Bound*, uma abordagem diferente para resolver o mesmo problema. Em melhoria a primeira abordagem, esse algoritmo pode ser utilizado diretamente, chegando a solução ótima, sem a necessidade de uma atribuição inicial. Nos trabalhos citados, os resultados experimentais mostram que os tempos computacionais crescem vertiginosamente com o tamanho da instância. O autor ainda ressalta que o método utilizado por ele encontra a solução ótima em estágios prematuros de execução e muito tempo é gasto provando que esta solução é ótima.

Bozer e Carlo [7] também estudaram a atribuição de docas aos caminhões. Os autores consideram um CCD, com número igual de docas de entrada e saída, na qual o transbordo de cargas é feito à noite, podendo a atribuição mudar de uma noite para a outra. Eles propõem inicialmente uma formulação baseada no problema de atribuição quadrática retilínea para minimizar o esforço de manipulação de materiais. Esta formulação é ineficiente para resolver problemas de grande escala, visto que é uma decisão de curto prazo a ser tomada todos os dias. Os autores então propõem uma heurística baseada em *simulated annealing*, que chega à solução ótima em problemas pequenos e apresenta desempenho superior a heurísticas de troca de pares existentes na literatura para problemas grandes. O impacto do formato da estação sobre a movimentação de materiais também é analisado, chegando à conclusão que estações mais planas requerem menos esforços.

K. Gue em [12] existe uma mudança de foco do trabalho, sua análise baseou na alocação de docas examinando o efeito que um sequenciamento planejado ($look-ahead-scheduling$) tem sobre o fluxo de materiais e no layout de um centro. O sequenciamento planejado faz oposição ao método "primeiro chegar - primeiro sair", dessa forma, observa-se todos os elementos de um horizonte definido e não apenas o próximo. Assim, para determinar o *layout* menos trabalhoso, o autor propõem a procura da solução no espaço de soluções possíveis com auxílio do algoritmo de busca local. O custo do *layout* pode ser determinado se o fluxo de material resultante é conhecido. É exposto um algoritmo específico chamado busca futura ($look-ahead$), que testa as soluções por meio de simulação. Os resultados indicam ser possível

economizar entre 15% a 20 % nos custos.

O problema se concentra sobre a sucessão dos caminhões de chegada e saída nas docas de um terminal de *crossdocking*, ou seja, onde e quando os veículos devem ser processados. Alguns trabalhos científicos cujo foco se encontra em problemas relacionados ao sequenciamento de caminhões para CCD são citados por Boysen e Fliedner [6] e encontram-se resumidos na Tabela 2.1 adaptada pela autora, que acrescentou os trabalhos produzidos por Lira [16], Lima [15], e por Araújo e Melo [3].

Tabela 2.1: Histórico de pesquisas relacionadas ao sequenciamento de caminhões em CCD (Fonte: Adaptado de Boysen e Fliedner [6]).

Publicação	Notação do Problema	Complexidade	Contribuição
McWilliams et al. [17]	$[E\|t_{io}\|C_{max}]$	Desconhecida	HM, S
Boysen [5]	$[E\|p_j = p, no-wait, t_j = 0\|\sum T_o]$	Desconhecida	M, HM, E
Boysen et al.	$[E2\|p_j = p, change\|C_{max}]$	NP-difícil	M, HS, HI, E, P
Chen e Lee [8]	$[E2\|t_j = 0\|C_{max}]$	NP-difícil	B, E, P
Chen e Lee [8]	$[E2\|t_j = 0, p_j = p\|C_{max}]$	NP-difícil	P
Chmielewski	$[EM\|r_j, d_j, limit, doors, t_io\|*]$	Desconhecida	M, E
Miao et al. [18]	$[M\|limit, t_{io}\|*]$	NP-difícil	M, HM, P
Boysen [5]	$[M1\|p_j = p, t_j = 0, change\|\sum S_p]$	NP-difícil	M, HM, E, P
Yu e Egbelu [23]	$[E2\|change\|C_max]$	Desconhecida	M, HS
Chen e Song [9]	$[E\|t_j = 0\|C_{max}]$	NP-difícil	M, HS, B
Boysen e Fliedner [6]	$[E\|t_{io}, fix\|\sum w_s U_s]$	NP-difícil	M, P
Boysen e Fliedner [6]	$[E\|t_i = 0, fix\|\sum w_s U_s]$	NP-difícil	P
Lira [16]	$[F2(P)\|CD\|C_{max}]$	NP-difícil	HM
Lima [15]	$F2\|CD\|\sum C_j^2$	NP-difícil	M, HS, HM
Araújo e Melo [3]	$[F2(P)\|CD\|C_{max}]$	NP-difícil	H

As siglas da coluna "Contribuição"correspondem a: M (Modelo Matemático),B (Programação por *bound*), HI (Procedimento de Melhoria da Heurística), HS (Começa Heurística para solução inicial), HM (Metaheurística), P (Propriedades e.g. complexidade do problema), S (Abordagem por Simulação) e E (Procedimento de resolução exato)

Em [6] também é apresentada uma classificação de trabalhos. Tal classificação foi feita baseada principalmente em um esquema sobre problemas determinísticos de sequenciamento de caminhões. Foi utilizada uma notação em tuplas, que é aplicada em sequenciamento de máquinas e problemas de filas. Como os problemas de *crossdocking* apresentam peculiaridades adicionais, como a atribuição de cargas a veículos de saída, a classificação proposta pelos autores inclui também atributos específicos desse tipo de problema, relacionando-se com a forma de trabalhar das do-

cas, as características operacionais e os objetivos que guiaram a otimização tratados por $\alpha|\beta|\gamma$, respectivamente. A estudo realizado por Belle et al. [4] também aborda a classificação utilizada em [6], entretanto seu foco principal está na classificação por características dos centros.

McWilliams et al. [17] apresentaram o primeiro trabalho focado em sequenciamento de caminhões com um horizonte curto de planejamento. Os autores consideram um CCD com docas que podem ser de entrada ou saída, para a consolidação de carga para a indústria de entrega de encomendas no serviço postal. O centro em estudo, tinha as encomendas descarregadas e encaminhadas aos caminhões de despacho por meio de uma rede de correias transportadoras. Nenhum tipo de armazenamento foi considerado. Busca-se no trabalho selecionar docas de recebimento de forma a não sobrecarregar os classificadores, mas maximizar taxa de transferência, com o objetivo final de minimizar o *makespan*. A solução foi abordada por meio de um algoritmo genético, o qual interage com um modelo de simulação. Em termos de notação em tuplas, o modelo está apresentado na Tabela 2.1. Sendo que β significa que todos os jobs tem o mesmo tempo de processamento, não podem aguardar em estoques intermediários e os tempos de transbordo para cada par de docas é dado como parâmetro.

Uma estrutura de *crossdocking* com aplicações reais, entretando específica é apresentada por Boysen [5]. Seu trabalho aborda a alocação de docas a caminhões que chegam ao CCD no caso especial da cadeia suprimentos alimentícia, onde nenhum tipo de armazenamento temporário é permitido dentro do centro não climatizado. As docas são segregadas entre recebimento e despacho e devem ser alocadas de acordo com uma sucessão cronológica discreta. Assim, o autor apresenta uma formulação, cujo o objetivo é minimizar o tempo de fluxo, o tempo de processamento e o atraso dos caminhões de saída. O autor dá ênfase a programação dinâmica associada a um grafo direto acíclico. O caminho mínimo nesse grafo corresponde a solução ótima do problema tratado, que consiste em sequenciar caminhões de chegada e saída em um terminal de crossdocking. Também é apresentada uma heurística baseada em *simulated annealing* para limites próximos ao valor ótimo para instâncias reais, representadas por mais de 25 caminhões de chegada.

Chen e Lee [8] abordam o problema do sequenciamento dos caminhões de forma análoga ao problema clássico de *flow shop* com duas máquinas (F2||Cmax), cujo objetivo é minimizar o *makespan*, o problema trata as restrições de *crossdocking*, isto é, o carregamento de um caminhão só pode ser iniciado se todos os produtos necessários para aquele caminhão tiverem sido descarregados na doca de entrada. Existe apenas uma doca de recebimento e uma doca de despacho, fazendo com que a primeira máquina representa a doca de entrada, a qual realiza a operação de

descarregamento dos caminhões; e a segunda máquina representa a doca de saída, aonde são realizadas as operações de montagem de caminhões contendo produtos com um mesmo destino. Os caminhões são representados por dois conjuntos distintos de *jobs*, os quais estão relacionados a cada uma das duas máquinas. Os autores comprovam que se trata de um problema NP-difícil e apresentam casos especiais em que ele pode ser resolvido polinomialmente. Ao final propõe um algoritmo de *branch and bound* para resolução do problema. Os resultados mostram que o problema pode resolver instâncias com até 60 caminhões em tempo aceitável.

Lima [15] trabalhou o problema tratado em Chen e Lee [8] entretanto considerou um função objetivo diferente. Em seu trabalho Lima [15] preocupou-se em minimizar a soma das datas de conclusão ponderadas de todos os *jobs*, o problema foi modelado como $F2|CD|\sum C_j^2$. É proposto um modelo de programação inteira com formulação indexada no tempo, que é resolvido utlizando o método de relaxação lagrangeana e geração de colunas, são propostas ainda duas heurísticas de forma a tentar obter limites mais próximos da solução ótima do problema.

Chen e Song [9] estendem o trabalho de Chen e Lee [8] considerando múltiplas docas de entrada e saída, denominando problema de *crossdocking* híbrido de dois estágios, ou seja, com várias máquinas em cada estágio, paralelas e idênticas, uma para cada porta do CCD, a qual representa uma situação logística mais realística. Propõe-se primeiramente um modelo de programação linear inteira mista que resolve instâncias de pequena escala, o qual é resolvido pelo CPLEX. Em seguida, quatro heurísticas construtivas são propostas para instâncias de média e grande escala: JRH (*Johnson's rule-based heuristics*), JLPTH (*Johnson's rule-based LPT heuristics*), DJRH (*Dynamic Johnson's rule-based ready time heuristic*) e DJLPTH (*Dynamic Johnson's rule-based LPT heuristic*). Essas heurísticas foram baseadas no algoritmo de *Johnson* [13]. O artigo introduz o conceito de instâncias auxiliares para trabalhar com essas heurísticas, o qual elimina as restrições de *crossdocking* e adapta os parâmetros de entrada de forma que seja possível ordenar os *jobs* para cada estágio. A diferença entre JRH e JLPTH ou entre DJRH e DJLPTH está na regra que determina a ordem do sequenciamento durante o segundo estágio: a primeira ordena em ordem decrescente dos *ready times* e a segunda conforme a regra do maior tempo de processsamento (*largest processing time* - LPT). As duas outras heurísticas se utilizam do algoritmo de *Johnson* de forma dinâmica, assim a instância original é continuamente atualizada para uma nova que considera somente os *jobs* que ainda não foram processados. O artigo, através da realização de experimentos computacionais, conclui que a heurística JLPTH apresentou os melhores resultados.

Com base nos trabalhos científicos de Chen e Lee [8] e Chen e Song [9], Araújo e Melo [3] analisam e avaliam heurísticas já existentes na literatura para o problema em

questão e propõe outras, objetivando determinar aquelas que obtêem melhores *GAPs* quando comparadas a um limite inferior. Duas heurísticas propostas em [9] são implementadas e comparadas à outros cinco pares de heurísticas construtivas propostas, sendo que cada par é diferenciado de acordo com a regra de reordenação do segundo estágio, ora por *ready times* ora por LPT. Neste último caso, quando mais de um *job* está pronto para ser processado simultaneamente, é escolhido primeiro o *job* com maior tempo de processamento (LPT). Esses cinco pares são detalhados a seguir:

- LPT1R2: ordenação dos *jobs* no primeiro estágio em ordem decrescente.
- JRHm: Lee e Vairaktarakis [14] realizaram um estudo de heurísticas construtivas para a minimização do *makespan* em *flowshops* híbridos, no qual se propõe uma heurística para o caso particular com apenas dois estágios. Ao calcular o tempo de processamento de cada *job* nos estágios, divide-se o resultado pelo fator número de máquinas no estágio. As heurísticas de [9] foram modificadas de acordo com este critério de forma que o cálculo dos *jobs* fictícios passou a possuir vínculo com o número de máquinas em cada estágio.

As três próximas heurísticas foram embasadas na heurística NEH, a qual foi proposta por Nawaz et al. [19] e é bem conhecida e consagrada por apresentar resultados bem interessantes para os problemas clássicos de *flowshop* permutacional sem grande esforço computacional. Esses três pares de heurísticas utilizam uma lista de inserção e, a cada passo, avaliam de alguma forma o *makespan*, mesmo sendo ele fictício ou para um dos estágios apenas.

- BDNEH: baseia-se na construção de uma ordem de inserção para os *jobs* do segundo estágio, obtida pela ordenação decrescente da soma dos tempos de processamento com os tempos de processamento de seus precedentes.
- BDm1: a diferença dessa heurística em relação a BDNEH se encontra na definição da ordem de inserção para os *jobs* do segundo estágio, uma vez que se considera também o número de máquinas no primeiro estágio.
- NEHtp2: essa heurística se diferencia da BDNEH no momento de Inserção para definição da sequência preliminar do segundo estágio. Além de considerar os tempos de processamento neste estágio, soma-se um valor médio do tempo de processamento dos seus precedentes no primeiro estágio dividindo-se a soma dos tempos de processamento deles pelo número de máquinas no primeiro estágio.

O GAP mínimo das heurísticas, na média, pertence a NEHtp2LPT, enquanto que o melhor GAP dos piores casos das heurísticas pertence à JLPTH na maior

parte das vezes. Além disso, também se verifica um desempenho superior em todos os experimentos computacionais realizados da escolha de LPT em detrimento dos *ready time*.

Lira [16] também estudou o problema, em seu trabalho analisou buscas locais, a fim de, pesquisar o espaço de soluções, para propor uma metaheurística *Iterated Greedy* com busca local troca simples. Essa metaheurística encontrou soluções de boa qualidade em um tempo computacional viável.

Miao et al. [18] analisaram o problema de *crossdocking* com múltiplas docas, entretanto não há exclusividade para docas de recebimento e despacho. A viabilidade de uma solução era afetada por três fatores: a janela de tempo em que cada caminhão pode chegar ou deixar o centro, o tempo operacional para que haja o transbordo da carga e a capacidade total do terminal *crossdocking* que determina a quantidade de carga que pode ser temporariamente armazenada. Quando nenhuma doca foi encontrada com tempo suficiente para processar completamente um caminhão, este se torna uma tarefa perdida. Consequentemente, a função objetivo minimiza o custo de penalidade por tarefas perdidas. Em acréscimo, um termo adicional abrange o custo operacional influenciado pela localização das portas e a distância a ser transposta pelas empilhadeiras. Com o aumento da instância uma formulação inteira expande rapidamente, não sendo viável uma solução ótima. Os autores propõe uma busca Tabu e um algoritmo genético capazes de resolver o problema. A busca Tabu especificamente demonstrou ser bastante eficiente em um estudo computacional com diferentes tamanhos de instâncias comparados ao método tradicional por CPLEX.

Yu e Egbelu [23] propõe um modelo para o problema considerando apenas uma doca para descarregamento e outra para montagem dos caminhões. Além disso, o modelo considera uma área de estoque temporário em frente à doca de montagem. O objetivo do artigo é encontrar o melhor sequenciamento de caminhões no CCD de forma a minimizar o tempo de operação total do *crossdocking*. As atribuições relacionadas ao produto, tanto para os caminhões de entrada quanto de saída, são determinadas simultaneamente com o sequenciamento de caminhões. Assim, são propostas três diferentes abordagens para resolução do problema. Na primeira abordagem, desenvolveu-se um modelo matemático com o objetivo de minimizar o *makespan*. A segunda abordagem utiliza enumeração completa para gerar todas as possíveis sequências de caminhões para o problema. Essas duas abordagens se mostraram ineficientes para a resolução de problemas de média e grande escala devido ao elevado tempo computacional que requerem. Assim, uma terceira abordagem foi proposta empregando um algoritmo heurístico que se mostrou eficiente, mas que não garante encontrar a solução ótima para o problema.

Um problema que difere das abordagens dos trabalhos anteriores é apresentado por Alpan et al. [2]. Normalmente, nos estudos de sequenciamento, a função objetivo está relacionada com uma função do tempo, como makespan ou tempo de atraso. Neste trabalho, Alpan et al. [2] focam o custo operacional de manter estoques temporários de mercadorias dentro do CCD confrontado com custo de interromper um carregamento e fazer a troca de caminhões. O recebimento de caminhões utiliza política $FIFO$ (do inglês *first in first out*, primeiro que entra, primeiro que sai) nas múltiplas portas e o resultado busca otimizar apenas a sequência de caminhões de despacho que devem estar presentes nas docas em cada intervalo de tempo discretizado, tal que, o custo de estoque e o custo de preempção seja minimizado. O transbordo da carga é feito em unidades de tempo idênticas, sendo que um *pallet* pode ser diretamente transferido de caminhões ou armazenado pagando seu custo. O estudo considerou que há trabalhadores e máquinas suficiente para carga e descarga de todos os caminhões ao mesmo tempo. As características apresentadas são condizente com terminais *crossdocking* em fábricas de manufatura, em que o terminal é posicionado ao final das linhas de produção e a sequência de recebimento é imposta pelo plano de produção. A solução foi avaliada e utilizando progrmação dinâmica, onde é feito um estudo detalhado dos limites do espaço de solução.

A dissertação em questão apresenta primeiramente um modelo matemático indexado no tempo para resolução de instâncias de pequeno porte, esse modelo é comparado ao modelo presente na literatura [9]. Posteriormente trata as duas heurísticas propostas por Chen e Song [9] que apresentaram melhores resultados e as duas heurísticas propostas por Araújo e Melo que também alcançaram melhores resultados. Assim melhorias são propostas para as heurísticas de Araújo e Melo, que são comparadas às heurísticas da literatura [9].

2.3 Definição do Problema

Consideramos um centro de *crossdocking* no qual os veículos que chegam carregados com mercadorias vindos de vários fornecedores diferentes, são direcionados a uma das docas de entrada, onde os produtos são descarregados. Os veículos que exercem essa função recebem o nome de caminhões de entrada. Cada caminhão de entrada pode conter vários produtos diferentes, vindos de um único fornecedor. Os caminhões que transportam os pedidos do CCD aos clientes são chamados caminhões de saída. Esses caminhões são direcionados as docas de saída, onde cada pedido é carregado, uma vez que, o caminhão está carregado o pedido é enviado ao cliente. O início do carregamento de um caminhão de saída só pode ser iniciado após o descarregamento de todos os caminhões de entrada dos quais esse depende. Cada caminhão de saída

só pode deixar o CCD após o seu carregamento ter sido concluido. É importante ressaltar que o problema considera a existência de mais de uma doca (processador) tanto de entrada quanto de saída e que o tempo de processamento para descarregar e carregar cada caminhão é conhecido. Assim o objetivo do problema envolve além do sequenciamento de caminhões o sequenciamento das docas de forma a minimizar o *makespan*, momento em que o último veículo de saída é expedido. O problema é ilustrado pela Figura 2.3.

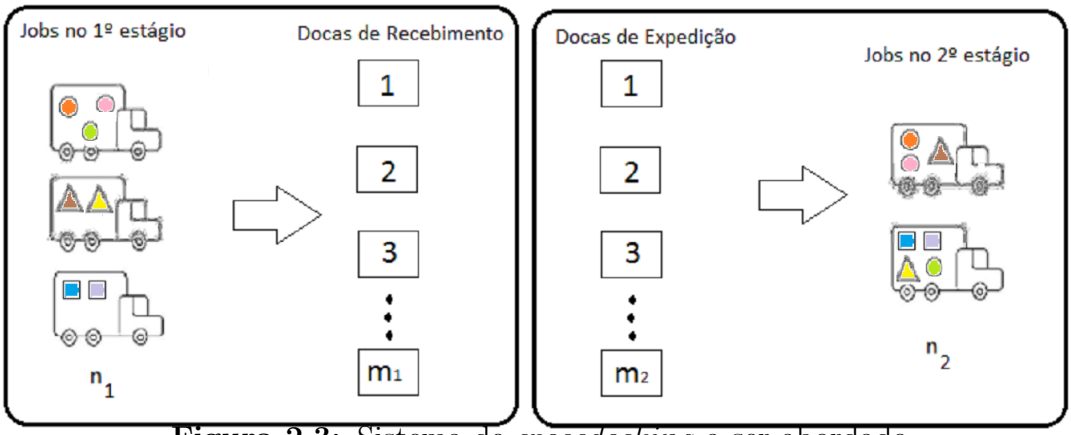

Figura 2.3: Sistema de *crossdocking* a ser abordado.

Chen e Lee em [8] estudaram o problema do tipo $F2|CD|C_{max}$, um problema de $flowshop$ com duas máquinas, que possui restrições de *crossdocking* e objetiva minimizar o *makespan*. Esse problema é comprovado como NP-difícil [8]. Chen e Song em [9] generalizaram o estudo de Chen e Lee [8] com um artigo sobre o problema de $flowshop$ com máquinas paralelas, o qual também é conhecido como $flowshop$ híbrido e que é um problema, segundo Garey e Johnson [11], NP-difícil no sentido estrito. O que justifica o uso de abordagens heurísticas, bem como a combinação dessas abordagens ao método exato para resolver o problema.

Dessa maneira, o problema tema deste trabalho pode ser representado por $F2(P)|CD|C_{max}$. Um problema análogo ao $flowshop$ híbrido de dois estágios ("P"indica que em cada estágio existem multiprocessadores idênticos em paralelo), com restrições de *crossdocking*, no qual a função objetivo é minimizar o *makespan*.

Capítulo 3

Formulações Matemáticas

Apresentaremos neste capítulo as definições e a formulação geral do problema estudado, posteriormente explicitaremos o modelo matemático proposto por Chen e Song em [9] seguido do modelo de programação inteira mista indexado no tempo proposto neste trabalho e algumas considerações.

3.1 Definições e Formulação

Uma vez que, a intenção é comparar os dois modelos, o modelo aqui proposto seguiu a mesma metodologia utilizada na modelagem proposta por Chen e Song em [9], considerando a notação mais próxima possível.

Índices:

i: estágio de processamento, $i \in I = \{1, 2\}$

v_{ia_i}: processador no estágio i, $v_{ia_i} \in V_i = \{v_{i1}, ..., v_{im_i}\}, a_i = 1, ..., m_i$

j_{ie_i}: job a ser processado no estágio i, $j_{ie_i} \in J_i = \{j_{i1}, ..., j_{in_i}\}, e_i = 1, ..., n_i$.

Parâmetros de entrada:

m_i: número de processadores(docas) no estágio i.

n_i: número de $jobs$ no estágio i.

p_{ie_i}: tempo de processamento no estágio i do job e_i

S_j: conjunto de subconjuntos de J_1 precedentes ao job $j_{2e_2} \in J_2$,$S_{e_2} \in S = \{S_1, ..., S_{n_2}\}$.

Variável de decisão:

C_{max} : tempo que se demora para processar todos os jobs nos dois estágios ($Makespan$).

3.2 Modelo Matemático proposto por Chen e Song (MCS)

O primeiro modelo apresentado foi proposto por Chen e Song em [9]. As variáveis específicas deste modelo são discriminadas abaixo:

Parâmetros de entrada:

Q: número grande, que deve ser pelo menos maior que o *makespan*. Na prática utilizou-se $Q = \sum_{i \in I} \sum_{j_{ie_i} \in J_i} p_{ie_i}$.

Variáveis de decisão:

C_{ie_i}: data de conclusão do *job* j_{ie_i} no estágio i;
$x_{ia_ie_i} = 1$, se o *job* j_{ie_i} é alocado ao processador v_{ia_i} no estágio i; senão $x_{ia_ie_i} = 0$
$y_{ie_if_i} = 1$, se o *job* j_{ie_i} precede o *job* j_{if_i} no estágio i; senão $y_{ie_if_i} = 0$

A partir da notação e variáveis de decisão acima, segue o Modelo de Programação Inteira Mista:

$$min \quad C_{max} \tag{3.1}$$

$$sujeito \quad a:$$

$$\sum_{v_{ia_i} \in V_i} x_{ia_ie_i} = 1; \quad \forall i \in I, j_{ie_i} \in J_i; \tag{3.2}$$

$$C_{1e_1} \geq p_{1e_1}; \quad \forall j_{1e_1} \in J_1; \tag{3.3}$$

$$C_{2e_2} \geq C_{1e_1} + p_{2e_2}; \quad \forall j_{2e_2} \in J_2, j_{1e_1} \in S_{e_2}, S_{e_2} \in S; \tag{3.4}$$

$$C_{ie_i} + Q(2 + y_{ie_if_i} - x_{ia_ie_i} - x_{ia_if_i}) \geq C_{if_i} + p_{ie_i};$$
$$\forall i \in I, j_{ie_i}, j_{if_i} \in J_i, v_{ia_i} \in V_i; \tag{3.5}$$

$$C_{if_i} + Q(3 - y_{ie_if_i} - x_{ia_ie_i} - x_{ia_if_i}) \geq C_{ie_i} + p_{if_i};$$
$$\forall i \in I, j_{ie_i}, j_{if_i} \in J_i, v_{ia_i} \in V_i; \tag{3.6}$$

$$C_{max} \geq C_{ie_i}; \quad \forall i \in I, j_{ie_i} \in J_i; \tag{3.7}$$

$$x_{ia_ie_i}, y_{ie_if_i} \in \{0, 1\}; \quad \forall i \in I, j_{ie_i}, j_{if_i} \in J_i, v_{ia_i} \in V_i; \tag{3.8}$$

A função objetivo do problema (3.1) visa minimizar o tempo necessário para o processamento de todos os *jobs* (*makespan*). O conjunto de restrições (3.2) garante que em cada estágio de processamento, cada *job* será atribuído a exatamente um

processador (ou doca), as restrições (3.3) indicam que cada *job* é processado no primeiro estágio, sendo que o tempo de conclusão dos *jobs* do primeiro estágio devem ser maiores ou iguais aos seus tempos de processamento. O conjunto (3.4) garante que cada *job* j_{2e_2} pertencente ao segundo estágio deve ter data de conclusão maior ou igual a data de conclusão dos seus precedentes acrescida de seus tempos de processamento, assim um *job* do segundo estágio somente inicia seu processamento após a conclusão de todos os seus *jobs* precedentes. Já as restrições (3.5) e (3.6) são as restrições que trabalham com a não interferência de *jobs*, ou seja, são as restrições disjuntivas do modelo que proíbem os *jobs* de serem processados simultaneamente em um mesmo processador. Para uma dada sequência de *jobs*, somente um conjunto entre (3.5) e (3.6) é ativado mas para isso os *jobs* j_{ie_i} e j_{if_i} devem ser designados ao mesmo processador. Por exemplo, se o *job* j_{ie_i} é processado antes do *job* j_{if_i} no mesmo processador v_{ia_i}, temos $C_{if_i} \geq C_{ie_i} + p_{if_i}$ e $x_{ia_ie_i} = x_{ia_if_i} = 1$ e $y_{ie_if_i} = 1$. A restrição(3.6) é satisfeita. Ao mesmo tempo, (3.5), também é satisfeita devido ao Q ser suficientemente grande. O conjunto (3.7) define o tempo máximo de conclusão de todos os *jobs*, e por fim, (3.8) específica os domínios das variáveis de decisão do modelo.

3.3 Modelo Matemático Indexado no Tempo (MIT)

Nos modelos indexados no tempo, o horizonte de tempo é dividido em períodos e, assim as variáveis de decisão são também indexadas em cada um dos períodos empregados. A abordagem leva a modelos, em geral, com um número maior de restrições e variáveis de decisão, entretanto, suas relaxações lineares fornecem limitantes muito mais fortes que outras formulações encontradas na literatura.

Na formulação indexada no tempo é preciso definir inicialmente um horizonte de tempo T, esse horizonte é discretizado em t = 0,1, 2, ..., T períodos. Em cada período t, é tomada a decisão se algum *job* inicia seu processamento em alguma máquina ou não. É importante ressaltar que dependendo do problema tratado essa discretização pode ser adaptada. No caso estudado faz-se necessário analisar cada unidade de tempo.

Lima [15], apresentou um modelo matemático indexado no tempo que considerou apenas uma doca de processamento de entrada e uma de saída, sendo o objetivo tratado a minimização do somatório dos tempos de conclusão das tarefas. Dessa maneira, o modelo indexado no tempo apresentado nessa dissertação se difere do modelo supracitado por considerar múltiplas docas de processamento em cada estágio, e a função objetivo avaliada é a minimização do *makespan*.

As variáveis específicas para o modelo proposto foram:

Parâmetros de entrada:

T: Horizonte de tempo considerado inicialmente, na prática utilizou-se $T = \sum_{i \in I} \sum_{j_{ie_i} \in J_i} p_{ie_i}$.

Variáveis de decisão:

r_{2e_2}: data de liberação do *job* j_{2e_2};

$x_{ie_it} = 1$, se o *job* j_{ie_i} começar a ser processado no instante t no estágio i, senão $x_{ie_it} = 0$.

A partir da notação e variáveis de decisão acima, segue o Modelo de Programação Inteira Mista Indexado no Tempo proposto:

$$min \quad C_{max} \tag{3.9}$$

$$sujeito \quad a:$$

$$\sum_{t=0}^{T-p_{ie_i}} x_{ie_it} = 1; \quad \forall i \in I, j_{ie_i} \in J_i; \tag{3.10}$$

$$\sum_{e_i \in J_i} \sum_{s=max\{0,t-p_{ie_i}+1\}}^{t} x_{ie_is} \leq m_i; \quad \forall i \in I, t \in T; \tag{3.11}$$

$$r_{2e_2} \geq \sum_{t=0}^{T} (t + p_{1e_1}) * x_{1e_1t}; \quad \forall j_{2e_2} \in J^2, j_{1e_1} \in S_{e_2}, S_{e_2} \in S; \tag{3.12}$$

$$\sum_{t=0}^{T-p_{2e_2}} t x_{2e_2t} \geq r_{2e_2}; \quad \forall j_{2e_2} \in J_2; \tag{3.13}$$

$$C_{2e_2} \geq p_{2e_2} + \sum_{t=0}^{T-p_{2e_2}} t x_{2e_2t}; \quad \forall j_{2e_2} \in J_2; \tag{3.14}$$

$$C_{max} \geq C_{ie_i}; \quad \forall j_{ie_i} \in J_i; \tag{3.15}$$

$$x_{ie_it} \in 0, 1; \quad \forall i \in I, j_{ie_i} \in J_i, t \in T, \tag{3.16}$$

$$r_{2e_2} \geq 0; \quad \forall j_{2e_2} \in J_2; \tag{3.17}$$

$$C_{max} \geq 0; \tag{3.18}$$

O conjunto de restrições (3.10) garante que cada *job* j_{ie_i} deve iniciar seu processamento em uma, e somente uma, data dentro do horizonte de planejamento T. As restrições indicadas por (3.11) garantem que um *job* j_{ie_i} não inicia seu processa-

mento enquanto outro estiver sendo processado, da seguinte forma: para cada data $t \in T$, a restrição verifica se algum *job* j_{ie_i} começou a ser processado em uma data maior ou igual a $t - p_{ie_i} + 1$. Caso afirmativo, j_{ie_i} ainda está sendo processado, o número de *jobs* que podem ser processados ao mesmo tempo é menor ou igual ao número de máquinas existente naquele estágio de processamento considerado. O conjunto (3.12) são as restrições de precedência do tipo *crossdocking*. Assim, para cada relação de precedência, cria-se uma restrição neste conjunto, na qual, a data de liberação de cada *job* do segundo estágio, j_{2e_2}, deve ser maior ou igual a data de conclusão da tarefa precedente. Se o *job* possui mais de um precedente, prevalece aquela relativa ao *job* precedente com maior data de conclusão. Já o conjunto (3.13) garante que cada *job*, j_{2e_2}, do segundo estágio só inicia seu processamento depois de sua data de liberação. O conjunto (3.14) trata as datas de conclusão dos *jobs* do segundo estágio, assim garante que a data de conclusão de cada *job* do segundo estágio deve ser igual ao tempo de início de processamento do *job* acrescido do seu tempo de processamento. O conjunto de restrições (3.15) definem a função objetivo do modelo, indicando que a variável C_{max}, *makespan*, deve ser a maior data de conclusão, considerando todos os *jobs* processados. Por fim os conjuntos (3.16) à (3.18) definem o domínio das variáveis do modelo. A função objetivo é dada por (3.9) e visa minimizar o tempo necessário para o processamento de todos os *jobs* (*makespan*), como no modelo anterior.

A modo de exemplificação do modelo indexado no tempo, a Figura 3.1 ilustra uma solução viável à uma pequena instância onde todas as variáveis utilizadas no modelo são identificadas:

Observa-se primeiramente que existem duas máquinas paralelas em cada estágio de processamento, ou seja, duas docas de entrada e duas de saídas. No primeiro gráfico da Figura 3.1, referente ao primeiro estágio de processamento dos *jobs* é possível verificar que no instante t=0, dois *jobs* são iniciados j_{12} e j_{14} dessa maneira $x_{120} = 1$ e $x_{140} = 1$, já o *job* j_{11} inicia seu processamento no período t=1 por isso $x_{111} = 1$, e assim por diante. O mesmo acontece para o segundo gráfico, representante do segundo estágio de processamento, os *jobs* j_{21} e j_{22} iniciam seus processamentos em t=4 e por isso a variável x_{214} e $x_{224} = 1$. Os demais valores de x_{ie_it}, nos valores de t nos quais os *jobs* não são iniciados, são iguais a 0. Os tempos de processamento utilizados foram: p_{11}=2, p_{12}=1, p_{13}=2, p_{14}=4, p_{21}=4, p_{22}=3 e p_{23}=2. Considerando que existe precedência entre os *jobs*: $S_1 = \{j_{11}, j_{12}, j_{14}\}$, $S_2 = \{j_{11}, j_{14}\}$, $S_3 = \{j_{12}, j_{13}\}$. Diante dos dados, o tempo mínimo para o processamento de todos os *jobs*, ou seja, o *makespan* mínimo é igual a 9 unidades de tempo.

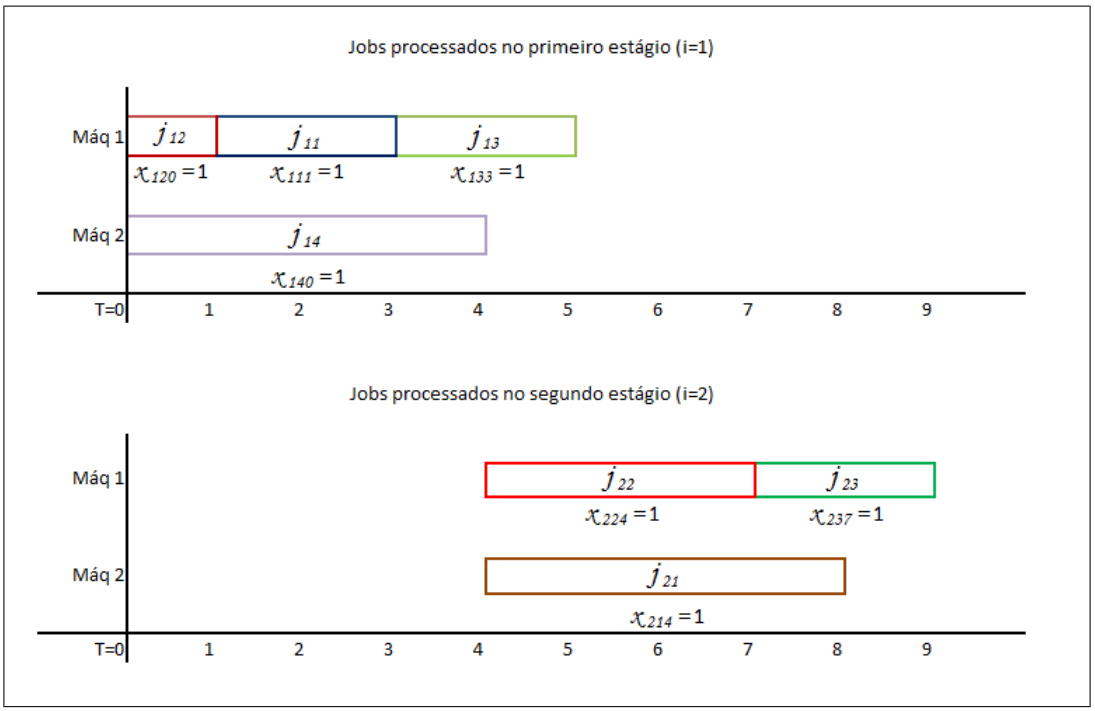

Figura 3.1: Abordagem de sequenciamento com indexação no tempo e múltiplas docas.

Capítulo 4

Experimentos Computacionais - Formulações

Os experimentos realizados neste trabalho foram executados utilizando-se o processador intel Core 2 Duo @ 2,26 GHz 2,27 GHz, 3,00 GB RAM, em sistema operacional Linux 32 bits, distribuição Ubunto 11.0.4. Foi utilizada a linguagem de programação AMPL e o software de otimização CPLEX 12.4.

4.1 Geração de Instâncias

As instâncias foram geradas através do gerador de números pseudo-aleatórios de $Mersenne\ Twister$, sendo que, as propriedades das instâncias geradas e aplicadas no modelo seguem abaixo:

1. O número de *jobs* no segundo estágio é gerado por uma distribuição uniforme entre 80% e 120% em relação ao número de *jobs* no primeiro estágio.

   $n_2 \sim Unif[0, x]$, sendo $x \sim [0, 8n_1, 1, 2n_1]$.

2. Quando o número de máquinas é determinístico tem-se:

   $m_1 = m_2$

   Já quando o número de máquinas é aleatorizado, gera-se o número de máquinas no primeiro e no segundo estágio seguindo uma distribuição uniforme entre o número máximo (maxmaqs) e o número mínimo (minmaqs) de máquinas dados como parâmetros de entrada:

   $m_1 \sim Unif[minmaqs, maxmaqs]$
   $m_2 \sim Unif[minmaqs, maxmaqs]$

3. O tempo de processamento dos *jobs* no primeiro e segundo estágio são gerados seguindo uma distribuição uniforme, que irá variar dependendo do teste realizado.

4. A matriz de precedências é do tipo binária (0 ou 1).

   $S_{n_2} \sim Unif[0,1]$;
   Cada linha da matriz é referente ao conjunto $S_{n_2} \in S = \{S_1, ..., S_{n_2}\}$;
   Essa matriz possui dimensão $n_2 \times n_1$. Se o valor for igual a 1, significa que o *job* do estágio 1 que se refere a essa coluna é precedente do *job* do segundo estágio que se refere a essa linha. Uma premissa considerada foi que os *jobs* do segundo estágio deveriam ter pelo menos um precedente.

4.2 Comparação dos modelos

Esta seção tem como objetivo comparar as formulações matemáticas. O modelo matemático com tempo indexado, proposto nesse trabalho, e o modelo proposto por Chen e Song [9] serão comparados por meio de suas relaxações lineares e do tempo de execução. O objetivo é avaliar se existe ou não uma dominância do modelo proposto sobre o modelo presente na literatura.

4.2.1 Metodologia dos experimentos computacionais

Para cada um dos modelos três grupos de máquinas foram analisados, sendo que dois deles utilizaram número de máquinas determinístico em cada estágio e um utilizou número de máquinas aleatório, como segue:

1. Duas máquinas nos dois estágios.

2. Quatro máquinas nos dois estágios.

3. Número de máquinas gerado por uma distribuição uniforme entre dois e quatro ($Unif\,[2,4]$) para cada estágio.

Dentro de cada um dos grupos de máquinas, variou-se o número de *jobs* no primeiro estágio ($n_1 = 5, 6, 7, 8$). Assim, foram consideradas 12 combinações distintas de número de máquinas e de *jobs*. Para cada uma dessas combinações 30 instâncias foram consideradas em um total de 360 instâncias testadas para cada modelo estudado. O tempo de processamento dos *jobs* seguiu uma distribuição uniforme entre dez e cem, ($p_{ie_i} \sim Unif[10, 100]$). A Tabela 4.3 apresenta um resumo das instâncias geradas.

Tabela 4.1: Variação das instâncias de teste: Resumimos como as instâncias variaram para os testes, a última coluna apresenta os valores de teste que foram considerados.

Parâmetro	Notação	Valores testados
Número de jobs no 1º estágio	n_1	5,6,7,8
Número de jobs no 2º estágio	n_2	[4,6] para n_1 igual a 5
		[4,7] para n_1 igual a 6
		[5,8] para n_1 igual a 7
		[6,9] para n_1 igual a 8
Número de máquinas no estágio i	m_i	2, 4, $Unif[2,4]$
Tempo de processamento	p_{ie_i}	$Unif[10,100]$

4.2.2 Resultados dos experimentos computacionais

Os resultados são expostos de forma resumida na Tabela 4.2, onde são apresentados resultados de Média, melhor caso e pior caso para os tempos computacionais e para os *Gaps*. Os tempos computacionais são apresentados tanto para a resolução do modelo na otimalidade (T(s)), quanto para a resolução do modelo relaxado (TR(s)). O *Gap* considerado é referente a distância entre o *makespan* encontrado pelo modelo e por sua relaxação linear, que é calculado pela seguinte fórmula:

$$Gap = \frac{Cmax(OTM) - Cmax(MR)}{Cmax(OTM)} * 100$$

$Cmax(OTM)$ é o *makespan* ótimo encontrado pelo modelo, e $Cmax(MR)$ é o *makespan* encontrado pelo modelo relaxado. É importante ressaltar que o tempo máximo de execução foi fixado em 3600 segundos, mas todas as instâncias de teste foram resolvidas antes de atingir esse limite de tempo.

Analisando a Tabela 4.2 é possível verificar que para o grupo que considera quatro máquinas e para o grupo com número aleatório de máquinas os problemas relaxados apresentaram valores de *Gap* iguais a zero no melhor caso, isso significa que, os problemas conseguiram atingir o valor ótimo para algumas instâncias do teste. Observa-se que com o aumento do número de máquinas os modelos passam a ser resolvidos em menor tempo computacional, e que os modelos relaxados apresentam valores menores de *Gap*, o que significa que a solução do modelo relaxado está mais próxima da solução ótima. Em contrapartida o aumento no número de *jobs* aumenta consideravelmente o tempo de execução dos modelos.

Comparando os modelos a partir da Tabela 4.2, nota-se que todos os *Gaps* pertencentes ao modelo indexado no tempo foram inferiores ou iguais aos *Gaps* resultantes do modelo de Chen e Song [9] relaxado linearmente. É importante deixar claro que para todas as 360 instâncias testadas o *Gap* apresentado pelo modelo

Tabela 4.2: Comparação dos modelos. Apresentamos melhor caso, média e pior caso para os três grupos de teste (2, 4, Unif[2,4] máquinas) de cada modelo completo, bem como para suas relaxações lineares. T (s) é o tempo computacional utilizado para resolver o modelo completo, dado em segudos; TR(s) é o tempo computacional gasto para resolver o problema relaxado, também apresentado em segundos.

m_i	n_1	n_2		Modelo indexado no tempo			Modelo de Cheng e Song [9]		
				T(s)	TR(s)	Gap(%)	T(s)	TR(s)	Gap(%)
2	5	[4,6]	Best	0,85	0,49	0,00	0,16	0,08	0,00
			Average	3,22	0,88	21,16	0,53	0,14	26,72
			Worst	10,79	2,01	36,58	2,07	0,20	41,63
	6	[4,7]	Best	1,17	0,48	11,43	0,21	0,13	16,46
			Average	17,49	1,39	26,49	2,31	0,16	34,64
			Worst	85,46	4,00	45,49	12,53	0,44	50,84
	7	[5,8]	Best	2,71	0,89	18,45	0,54	0,13	18,45
			Average	83,95	2,11	29,76	45,54	0,15	39,57
			Worst	1079,17	4,89	45,71	805,77	0,19	55,24
	8	[6,9]	Best	10,65	1,37	19,07	2,19	0,13	26,46
			Average	547,39	3,45	33,31	524,87	0,17	46,27
			Worst	3124,16	10,43	47,97	3595,99	0,45	58,78
Average 2 machines				163,01	1,96	27,68	143,31	0,15	36,80
4	5	[4,6]	Best	0,60	0,41	0,00	0,18	0,12	0,00
			Average	1,40	1,13	0,64	0,27	0,16	0,64
			Worst	2,68	3,29	8,02	0,72	0,31	8,02
	6	[4,7]	Best	0,88	0,48	0,00	0,18	0,12	0,00
			Average	2,51	1,15	4,67	0,69	0,15	4,67
			Worst	6,66	2,62	22,10	2,62	0,25	22,10
	7	[5,8]	Best	1,60	0,86	0,00	0,25	0,12	0,00
			Average	4,35	1,37	9,20	1,25	0,15	9,36
			Worst	15,25	2,35	24,50	2,97	0,41	24,50
	8	[6,9]	Best	2,52	1,00	0,00	0,27	0,12	0,00
			Average	20,68	1,77	13,73	44,43	0,16	15,24
			Worst	191,02	3,03	30,15	1014,45	0,50	50,15
Average 4 machines				7,23	1,35	7,06	11,66	0,16	7,48
Unif[2,4]	5	[4,6]	Best	0,49	0,34	0,00	0,18	0,12	0,00
			Average	2,11	0,84	12,71	0,36	0,14	15,25
			Worst	3,66	1,38	38,89	0,71	0,20	49,15
	6	[4,7]	Best	0,83	0,43	0,00	0,20	0,12	0,00
			Average	6,27	1,23	19,59	1,28	0,15	23,12
			Worst	45,98	3,28	45,71	9,33	0,42	45,71
	7	[5,8]	Best	1,99	0,67	0,00	0,28	0,12	0,00
			Average	23,20	1,83	23,62	8,27	0,17	29,53
			Worst	100,39	4,93	45,71	63,68	0,31	48,29
	8	[6,9]	Best	3,17	1,00	0,00	0,45	0,13	0,00
			Average	118,08	2,61	25,92	205,25	0,16	32,76
			Worst	1060,64	7,60	51,34	2664,99	0,22	51,63
Average [2,4] machines				37,42	1,63	20,46	53,79	0,15	25,17
Total Average				69,22	1,65	18,40	69,59	0,16	23,15

relaxado de Chen e Song [9] foram iguais ou superiores aos calculados pela relaxação linear do modelo indexado no tempo.

Em geral, para os grupos de teste considerados, os modelos apresentaram um tempo médio de execução similar. Apesar do modelo indexado no tempo relaxado ter apresentado um tempo de execução um pouco maior as instâncias foram resolvidas muito rapidamente, não chegando a 2 segundos na média. Já o limite de relaxação linear do modelo proposto foi em média 5 % melhor do que o do modelo apresentado na literatura, para as instâncias de teste consideradas. Diante dos resultados observados, torna-se importante analisar se existe uma relação de dominância do MIT sobre o modelo presente na literatura, como segue na seção seguinte.

4.3 Relação de dominância entre os modelos

Esta seção tem como objetivo avaliar se o modelo indexado no tempo possui uma formulação mais forte do que a formulação empregada no modelo de Chen e Song [9], para o problema estudado. Para isso, temos que demonstrar duas situações:

1. Demonstrar que qualquer solução viável e factível do MIT relaxado pode ser transformada em uma solução viável do MCS;
2. Demonstrar que existem soluções do MCS relaxado que não estão no conjunto de soluções do MIT relaxado, provando que a relaxação linear do MIT está mais próximo da envoltória convexa.

Proposição 4.3.1 *O modelo indexado no Tempo, proposto nesse trabalho, possui formulação mais forte que o modelo proposto por Chen e Song em [9] para o problema* $F2(P)|CD|C_{max}$.

Demonstração Qualquer solução x_{ie_it} da relaxação linear do MIT com custo z pode ser convertida em solução C_{ie_i} da relaxação linear do MCS com o mesmo custo definido por:

$$C_{ie_i} = \sum_{t=0}^{T-p_{ie_i}} (t + p_{ie_i}) x_{ie_it}, \qquad \forall j_{ie_i} \in J_i; \tag{4.1}$$

Como x_{ie_it} satisfaz o conjunto de restrições de sequenciamento dos *jobs* em (3.10) e (3.11), C_{ie_i} satisfaz o sequenciamento através dos conjuntos de restrições (3.2), (3.3), e (3.4). Já as restrições de *crossdocking* são atendidas pelo conjunto de restrições (3.12), (3.13) e (3.14) para x_{ie_it}, e pelas restrição (3.5), (3.6) para C_{ie_i}.

Por fim, as restrições (3.15) e (3.7) definem o *makespan* da formulação indexado no tempo e da formulação proposta por de Chen e Song [9] respectivamente.

Para a prova da segunda parte consideraremos uma instância do problema $F2(P)|CD|C_{max}$, que possui duas máquinas paralelas em cada estágio de processamento, ou seja, duas docas de entrada e duas de saídas ($m_1 = m_2 = 2$), 3 *jobs* a serem processados no primeiro estágio ($n_1 = 3$), e 2 *jobs* no segundo ($n_2 = 2$), com tempos de processamento dos *jobs* iguais à: $p_{11} = 29$, $p_{12} = 45$, $p_{13} = 45$, $p_{21} = 17$ e $p_{22} = 39$, com as seguintes precedências: $S_1 = \{j_{13}\}$ e $S_2 = \{j_{11}, j_{12}, j_{13}\}$. Essa instância apresenta solução ótima com *makespan*, $z = 111$, e com os *jobs* sequenciados como segue na figura 4.1.

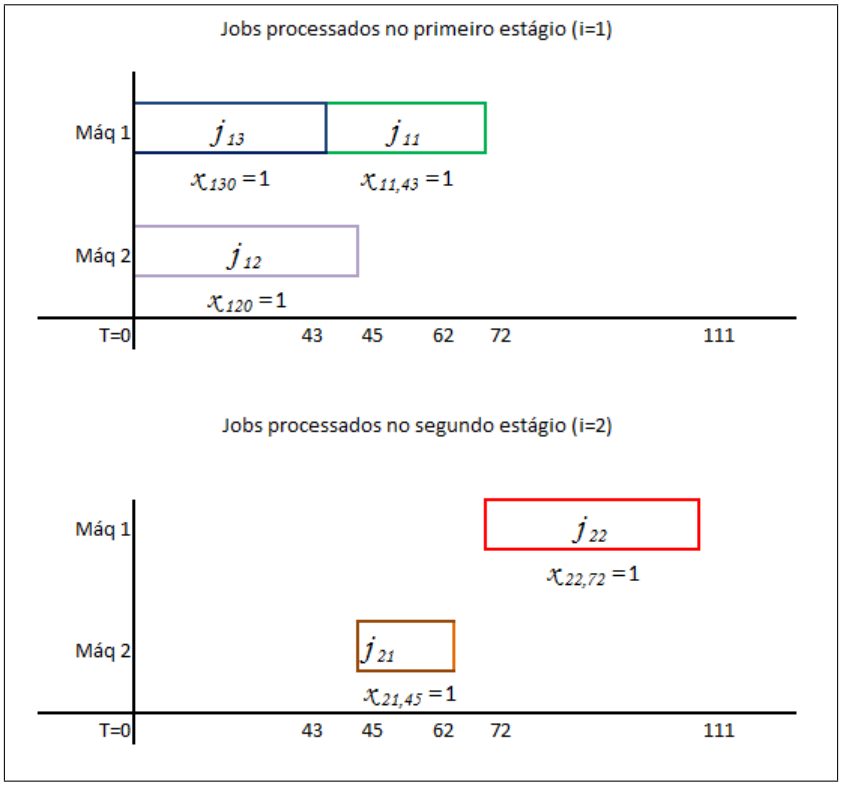

Figura 4.1: Sequência ótima, $z = 111$.

A solução encontrada pelo MCS relaxado linearmente, possui *makespan* igual à 84. Nessa solução os *jobs* iniciam seu processamento em t=0, assim, os tempos de conclusão dos *jobs* são $C_{11} = 29$, $C_{12} = 45$ e $C_{13} = 43$, onde $C_{1e_1} = p_{1e_1}$, da mesma forma, $C_{21} = 60$ e $C_{22} = 84$, $C_{2e_2} = C_{1e_1} + p_{2e_2}$. Esse fato ocorre devido à relaxação linear da variável $y_{ie_if_i}$, que trata a relação de precedência entre os *jobs* de um mesmo estágio, permitindo que a constante Q, desative o conjunto de restrições de *crossdocking*, restrição (3.5), (3.6). Já o MIT apresenta *makespan* igual à 91. ∎

4.4 Limites computacionais do Modelo Indexado no Tempo

Os experimentos computacionais realizados nesta seção objetivaram analisar os casos relacionados a pequenos CCD. Para o teste três grandes grupos de máquinas foram estudados:

1. Duas máquinas nos dois estágios.
2. Quatro máquinas nos dois estágios.
3. Seis máquinas nos dois estágios.

Dentro de cada um dos grupos de máquinas, variou-se o número de *jobs* no primeiro estágio ($n_1 = 5, 8, 10, 15$). Assim, foram consideradas 12 combinações distintas de número de máquinas e de *jobs*. Para cada uma dessas combinações 20 instâncias foram consideradas em um total de 240 instâncias testadas para cada grupo de tempo de processamento. Dois grupos com tempos de processamento diferentes foram analisados, com o objetivo de verificar a influência dos mesmo no tempo de execução do modelo, no primeiro grupo os tempos de procesamento dos *jobs* variou de maneira uniforme entre um e dez ($Unif[1, 10]$), já no segundo grupo o tempo de processamento dos *jobs* variou de maneira uniforme entre um e cinquenta ($Unif[1, 50]$). A Tabela 4.3 apresenta um resumo das instâncias geradas. É importante ressaltar que os conjuntos de instâncias são independentes dos conjuntos utilizados no experimento anterior. O tempo de execução dos teste foi fixado em 3600 segundos (1 hora).

Tabela 4.3: Variação das instâncias de teste: Resumimos como as instâncias variaram para os testes, a última coluna apresenta os valores de teste que foram considerados.

Parâmetro	Notação	Valores testados
Número de jobs no 1º estágio	n_1	5,8,10,15
Número de jobs no 2º estágio	n_2	[4,6] para n_1 igual a 5 [6,9] para n_1 igual a 8 [8,12] para n_1 igual a 10 [12,18] para n_1 igual a 15
Número de máquinas no estágio i	m_i	2,4,6
Tempo de processamento	p_{ie_i}	$Unif[1, 10]$ e $Unif[1, 50]$

As Tabelas 4.4 a 4.6 contêm resultados para cada grande grupo de instâncias criadas. A Tabela 4.4 refere-se às instâncias com duas máquinas em cada estágio, variando-se o número total de *jobs* para cada um dos grupos de tempos de processamento. A Tabela 4.5 é referente aos resultados para quatro máquinas, por fim

a Tabela 4.6 para seis máquinas em cada estágio. As tabelas apresentam os tempos computacionais e a porcentagem de instâncias resolvidas. O tempo máximo de execução foi fixado em 3600 segundos.

Analisando as três tabelas é verificável, como esperado, que a medida que o número de *jobs* aumenta o tempo computacional aumenta, entretanto a medida que o número de máquinas aumenta esse tempo diminui. Agora com relação aos tempos de processamento a medida que esse tempo aumenta o problema demora mais para ser resolvido, devido ao modelo ser indexado no tempo e o número de variáveis aumentar considerávelmente. É importante ressaltar que as instâncias de teste para cada uma das tabelas foi a mesma alterando somente o número de máquinas.

A partir da Tabela 4.4 verificamos que o modelo demora para resolver instâncias com dez *jobs* no primeiro estágio, o que fica mais difícil para quinze *jobs*, uma vez que, o modelo resolve apenas duas das vinte instâncias em 3600 segundos. Já na Tabela 4.5 encontramos resultados melhores que os encontrados para duas máquinas e piores que os encontrados para o grupo de seis máquinas.

Tabela 4.4: Tempos computacionais considerando 2 máquinas em cada estágio: Média, melhor caso, pior caso e porcentagem de instâncias resolvidas considerando 2 máquinas. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. A coluna ótimo apresenta a porcentagem de instâncias que foram resolvidas pelo modelo em 3600 segundos. Quando não é resolvido 100% das instâncias o pior valor anotado é o pior valor da instâncias resolvida.

			Grupo1		Grupo2	
n_1	n_2	Análise	Tempo(s)	Ótimo	Tempo(s)	Ótimo
5	[4,6]	Média	0,18	100%	7,18	100%
		Melhor Caso	0,00		0,00	
		Pior Caso	0,76		67,00	
8	[6,9]	Média	66,24	100%	66,93	100%
		Melhor Caso	0,38		5,02	
		Pior Caso	476,00		228,01	
10	[8,12]	Média	1179,468	80%	2093,95	60%
		Melhor Caso	3,49		55,01	
		Pior Caso	3315,12		3186,00	
15	[12,18]	Média	3331,65	10%	3600	0%
		Melhor Caso	306,00		3600	
		Pior Caso	1527,00		3600	

Tabela 4.5: Tempos computacionais considerando 4 máquinas em cada estágio: Média, melhor caso, pior caso e porcentagem de instâncias resolvidas considerando 4 máquinas. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. A coluna ótimo apresenta a porcentagem de instâncias que foram resolvidas pelo modelo em 3600 segundos. Quando não é resolvido 100% das instâncias o pior valor anotado é o pior valor da instâncias resolvida.

			Grupo1		Grupo2	
n_1	n_2	Análise	Tempo(s)	Ótimo	Tempo(s)	Ótimo
		Média	0,01		0,07	
5	[4,6]	Melhor Caso	0,00	100%	0,01	100%
		Pior Caso	0,02		0,19	
		Média	0,89		18,05	
8	[6,9]	Melhor Caso	0,02	100%	0,36	100%
		Pior Caso	6,89		225,00	
		Média	18,48		1299,69	
10	[8,12]	Melhor Caso	0,00	100%	4,03	75%
		Pior Caso	306,32		2722,02	
		Média	1473,65		3434,19	
15	[12,18]	Melhor Caso	37,59	70%	1147,03	10%
		Pior Caso	3522,01		2751,11	

Tabela 4.6: Tempos computacionais 6 máquinas em cada estágio: Média, melhor caso, pior caso e porcentagem de instâncias resolvidas considerando 6 máquinas. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. A coluna ótimo apresenta a porcentagem de instâncias que foram resolvidas pelo modelo em 3600 segundos. Quando não é resolvido 100% das instâncias o pior valor anotado é o pior valor da instâncias resolvida.

			Grupo1		Grupo2	
n_1	n_2	Análise	Tempo(s)	Ótimo	Tempo(s)	Ótimo
		Média	0,00		0,02	
5	[4,6]	Melhor Caso	0,00	100%	0,01	100%
		Pior Caso	0,00		0,03	
		Média	1,53		0,74	
8	[6,9]	Melhor Caso	0,01	100%	0,08	100%
		Pior Caso	29,18		5,35	
		Média	0,19		10,63	
10	[8,12]	Melhor Caso	0,00	100%	0,30	100%
		Pior Caso	1,02		116,79	
		Média	70,05		1828,42	
15	[12,18]	Melhor Caso	2,09	90%	2,45	60%
		Pior Caso	3574,00		3117,04	

4.5 Relaxação linear do modelo matemático

Nesta seção foram realizados testes com a finalidade de avaliar e comparar os resultados apresentados pelo modelo completo e suas Relaxações Lineares, que consiste em relaxar as condições de integralidade das variáveis de decisão.

Dentro de cada um dos três grupos de máquinas, variou-se o número de *jobs* no primeiro estágio ($n_1 = 5, 8, 10, 15$). Assim, foram consideradas 12 combinações distintas de número de máquinas e de *jobs*, igualmente como foi realizado no teste da Seção 4.4. Entretanto para cada uma dessas combinações 4 instâncias foram consideradas em um total de 48 instâncias testadas para cada grupo de tempo de processamento (grupo 1 tempos de processamento $Unif[1, 10]$ e grupo 2 com tempos $Unif[1, 50]$).

Os resultados obtidos pelo modelo completo foram comparados à três Relaxações Lineares parciais do modelo. O modelo completo considerou todas as restrições (3.10) à (3.18) citadas na Seção 3.3. A relaxação linear foi dividida em três relaxações parciais diferentes, sendo que na primeira proposta apenas a variável de sequenciamento referente ao primeiro estágio (x_{1e_1t}) foi relaxada linearmente. Na segunda proposta apenas as variáveis do segundo estágio (x_{2e_2t}) foram relaxadas linearmente. Por fim, x_{ie_it} foi realaxada, e dessa forma, nos dois estágios de processamento a variável resposta assumiu valores reais entre zero e um. Esse procedimento foi realizado com a finalidade de analisar a influência de cada um dos conjuntos de *jobs* no desempenho da formulação indexada no tempo no modelo proposto.

Após a execução do problema foram registrados o valor da função objetivo encontrado e o tempo de máquina. O tempo de execução foi limitado a 1 hora de processamento, caso a solução não seja encontrada em menos de 1 hora, a execução é interrompida, entretanto esses valores não foram consideredos nas tabelas dos resultados.

4.5.1 Resultados dos experimentos computacionais

Os resultados são expostos de forma resumida nas Tabelas 4.7 à 4.9, são apresentados resultados de Média, melhor caso e pior caso para os tempos computacionais (T(s))e para os *Gaps*, além da porcentagem de instâncias resolvidas (Resol). O *Gap* considerado é referente a distância entre o modelo completo e às relaxações lineares, que é calculado pela seguinte fórmula:

$$Gap = \frac{Cmax(MC) - Cmax(MR)}{Cmax(MC)} * 100$$

$Cmax(MC)$ é o *makespan* mínimo encontrado pelo modelo completo, quando o modelo não é resolvido em 3600 segundos a execução é interrompida e o valor de solução encontrado não é contabilizado para os cálculos, por isso nas tabelas de resultados alguns valores não foram alcançados. $Cmax(MR)$ o *makespan* mínimo encontrado pelos modelos relaxados, que algumas vezes também não chega a melhor solução em 3600 segundos.

Analisando as Tabelas 4.7 à 4.9 tem-se em alguns casos instâncias não resolvidas em 3600 segundos por isso quando o (-) é utilizado indica que nenhuma instância foi resolvida no tempo pré determinado, já quando um (*) é utilizado indica que a média não é real, pois algumas instâncias não foram resolvida e por sua vez não foram consideradas para os cálculos.

O leitor pode verificar que a relaxação linear apenas das váriaveis do primeiro estágio (x_{1e_1t}) apresentou limites mais fortes, ou seja, mais próximos da solução encontrada pelo modelo completo para todos os números de máquina. Entretanto, o tempo computacional gasto por esse problema relaxado é alto, sendo muitas vezes equivalente ao tempo gasto pelo modelo completo. Já a relaxação das variáveis do segundo estágio x_{2e_2t} apresentou resultados inferiores ao anterior, porém, em tempos computacionais mais viáveis. A relaxação total das variáveis apresentou os melhores tempos computacionais, todas as instâncias foram resolvidas instantaneamente, entretanto os limites encontrados foram inferiores aos apresentados pelas relaxações das variáveis separadamente.

A medida que o número de máquinas aumentou melhores resultados foram percebidos, na Tabela 4.9 todas as três relaxações apresentaram limites idênticos ou muito próximos dos obtidos pelo modelo completo.

Um fator importante a ser observado é que os *Gaps* não aumentaram em relação aos grupos de processamento, pelo contrário, na maioria dos casos os *Gaps* para os problemas do Grupo2 foram menores que os observados pelo Grupo1, lembrando que as instâncias consideraram as mesmas precedências alterando de um grupo para o outro apenas os tempos de processamentos. O que é um dos pontos positivos do modelo indexado no tempo proposto, uma vez que, o modelo completo encontrou muita dificuldade de resolver as instâncias do segundo grupo de processamento em tempo computacional viável.

Tabela 4.7: Resultados comparativos: Modelo completo(M.C) e modelos relaxados linearmente (R.L), considerando 2 máquinas em cada estágio. Apresentamos Média, melhor caso e pior caso dos tempos computacionais e *Gaps*. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. T(s) refere-se aos tempos de máquina, e o *Gap*(%) é a porcentagem de distância entre o valor encontrado pelo modelo completo e pelos modelos relaxados, a coluna Resol é referente a porcentagem de instâncias resolvidas em 3600 segundos. Em destaque estão os melhores valores de *Gap* encontrados entre as três relaxações.

2 máquinas em cada estágio												
			M. Completo			R.L de x_{1e_1t}			R.L de x_{2e_2t}			R.L de x_{ie_it}
n_1	n_2	Análise	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)
		Média	0,09	100%	8,07	0,06	100%	17,68	0,03	100%	26,61	0,01
5	U[4,6]	Melhor Caso	0,04		0,00	0,04		0,00	0,01		13,04	0,00
		Pior Caso	0,15		19,23	0,11		34,62	0,03		42,31	0,01
		Média	8,27	100%	11,39	4,61	100%	11,21	0,20	100%	27,74	0,01
8	U[6,9]	Melhor Caso	1,66		8,33	0,29		9,09	0,14		25,00	0,01
		Pior Caso	16,02		15,63	13,33		13,89	0,32		30,56	0,01
		Média	313,87*	50%	8,67	40,67*	50%	11,90	0,51	100%	32,66	0,02
10	U[8,12]	Melhor Caso	12,54		6,82	5,01		7,89	0,23		28,95	0,01
		Pior Caso	615,20		10,53	76,32		15,91	0,93		36,36	0,04
		Média	-	0%	-	-	0%	-	7,79	100%	-	0,28
15	U[12,18]	Melhor Caso	-		-	-		-	4,20		-	0,06
		Pior Caso	-		-	-		-	16,02		-	0,13
		Média	18,02	100%	4,21	12,96	100%	10,15	0,26	100%	17,57	0,03
5	U[4,6]	Melhor Caso	0,17		0,00	0,16		0,00	0,03		0,00	0,02
		Pior Caso	50,25		11,65	25,83		14,40	0,59		24,00	0,04
		Média	187,35	100%	6,43	964,08	100%	11,56	32,13	100%	26,27	0,49
8	U[6,9]	Melhor Caso	68,40		0,00	64,85		2,08	5,97		19,83	0,28
		Pior Caso	228,01		14,58	220,32		19,15	71,19		33,33	0,74
		Média	977,08*	50%	4,39	889,5*	50%	11,14	810,35	50%	31,08	0,71
10	U[8,12]	Melhor Caso	950,15		3,55	837,00		10,65	458,92		30,18	0,46
		Pior Caso	1004,00		5,23	942,00		11,63	1283,14		31,98	0,87
		Média	-	0%	-	-	0%	-	670,03*	25%	-	5,44
15	U[12,18]	Melhor Caso	-		-	-		-	670,03		-	2,27
		Pior Caso	-		-	-		-	-		-	9,71
Média Total			197,52		7,71	197,94		11,83	152,86		26,53	0,90

(-) Indica que as instâncias não foram resolvidas em 3600s.
(*) Indica que a média, não inclui as instâncias não resolvidas.

Tabela 4.8: Resultados comparativos: Modelo completo(M.C) e modelos relaxados linearmente (R.L), considerando 4 máquinas em cada estágio. Apresentamos Média, melhor caso e pior caso dos tempos computacionais e *Gaps*. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. T(s) refere-se aos tempos de máquina, e o *Gap*(%) é a porcentagem de distância entre o valor encontrado pelo modelo completo e pelos modelos relaxados, a coluna Resol é referente a porcentagem de instâncias resolvidas em 3600 segundos. Em destaque estão os melhores valores de *Gap* encontrados entre as três relaxações.

4 máquinas em cada estágio												
			M. Completo			R.L de x_{1e_1t}			R.L de x_{2e_2t}			R.L de x_{ie_it}
n_1	n_2	Análise	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)
		Média	0,01	100%	0,00	0,01	100%	6,58	0,01	100%	6,58	0,00
5	U[4,6]	Melhor Caso	0,00		0,00	0,00		0,00	0,00		0,00	0,00
		Pior Caso	0,01		0,00	0,01		26,32	0,01		26,32	0,00
		Média	0,18	100%	2,40	0,03	100%	1,19	0,03	100%	3,69	0,01
8	U[6,9]	Melhor Caso	0,02		0,00	0,01		0,00	0,01		0,00	0,00
		Pior Caso	0,54		4,76	0,06		4,76	0,07		10,00	0,01
		Média	78,13	100%	8,40	13,70	100%	12,78	0,14	100%	34,51	0,02
10	U[8,12]	Melhor Caso	0,44		0,00	0,30		0,00	0,09		29,17	0,01
		Pior Caso	306,32		9,09	51,26		20,83	0,21		54,55	0,02
		Média	2763,00*	25%	6,13	686,82*	25%	21,81	0,75	100%	38,63	0,04
15	U[12,18]	Melhor Caso	2763,00		6,00	686,82		15,63	0,58		31,25	0,02
		Pior Caso	-		6,25	-		28,00	0,94		46,00	0,05
		Média	0,17	100%	0,00	0,17	100%	0,00	0,05	100%	0,00	0,03
5	U[4,6]	Melhor Caso	0,06		0,00	0,02		0,00	0,03		0,00	0,00
		Pior Caso	0,19		0,00	0,17		0,00	0,08		0,00	0,02
		Média	3,38	100%	2,86	26,74	100%	1,43	0,63	100%	3,71	0,11
8	U[6,9]	Melhor Caso	0,31		0,00	0,62		0,00	0,09		0,00	0,06
		Pior Caso	10,82		8,99	104,20		5,81	2,04		8,99	0,18
		Média	1077,69	100%	4,20	772,79	100%	11,89	14,02	100%	16,08	0,16
10	U[8,12]	Melhor Caso	11,73		2,65	39,98		6,60	2,76		13,21	0,12
		Pior Caso	2417,19		6,54	134,00		15,53	26,17		18,58	0,21
		Média	-	0%	-	-	0%	-	897,51	100%	-	1,41
15	U[12,18]	Melhor Caso	-		-	-		-	495,44		-	0,20
		Pior Caso	-		-	-		-	1779,76		-	2,71
Média Total			351,06		3,25	96,36		8,53	134,23		16,25	0,22

(-) Indica que as instâncias não foram resolvidas em 3600s.
(*) Indica que a média, não inclui as instâncias não resolvidas.

Tabela 4.9: Resultados comparativos: Modelo completo(M.C) e modelos relaxados linearmente (R.L), considerando 6 máquinas em cada estágio. Apresentamos Média, melhor caso e pior caso dos tempos computacionais e *Gaps*. n_1 = número de *jobs* no estágio 1. n_2 = número de *jobs* no estágio 2. T(s) refere-se aos tempos de máquina, e o *Gap*(%) é a porcentagem de distância entre o valor encontrado pelo modelo completo e pelos modelos relaxados, a coluna Resol é referente a porcentagem de instâncias resolvidas em 3600 segundos. Em destaque estão os melhores valores de *Gap* encontrados entre as três relaxações.

6 máquinas em cada estágio												
			M. Completo			R.L de x_{1e_1t}			R.L de x_{2e_2t}			R.L de x_{ie_it}
n_1	n_2	Análise	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)	Resol	Gap	T(s)
		Média	0,00	100%	0,00	0,00	100%	0,00	0,01	100%	0,00	0,00
5	U[4,6]	Melhor Caso	0,00		0,00	0,00		0,00	0,01		0,00	0,00
		Pior Caso	0,00		0,00	0,01		0,00	0,01		0,00	0,00
		Média	0,06	100%	0,00	0,02	100%	0,00	0,00	100%	0,00	0,01
8	U[6,9]	Melhor Caso	0,01		0,00	0,01		0,00	0,00		0,00	0,01
		Pior Caso	0,15		0,00	0,04		0,00	0,00		0,00	0,01
		Média	0,73	100%	1,04	0,20	100%	5,90	0,09	100%	8,33	0,02
10	U[8,12]	Melhor Caso	0,05		0,00	0,01		0,00	0,06		0,00	0,01
		Pior Caso	1,02		4,17	0,42		12,50	0,13		16,67	0,03
		Média	64,92*	75%	6,58	6,53*	75%	15,67	0,45	100%	23,59	0,02
15	U[12,18]	Melhor Caso	12,80		4,00	2,13		12,50	0,29		20,83	0,01
		Pior Caso	109,37		8,33	9,01		18,52	0,38		25,93	0,03
		Média	0,03	100%	0,00	0,03	100%	0,00	0,01	100%	0,00	0,01
5	U[4,6]	Melhor Caso	0,01		0,00	0,01		0,00	0,01		0,00	0,00
		Pior Caso	0,03		0,00	0,04		0,00	0,01		0,00	0,01
		Média	0,19	100%	0,00	0,18	100%	0,00	0,07	100%	0,00	0,03
8	U[6,9]	Melhor Caso	0,11		0,00	0,09		0,00	0,03		0,00	0,02
		Pior Caso	0,29		0,00	0,36		0,00	0,10		0,00	0,05
		Média	0,84	100%	0,00	2,77	100%	0,00	0,26	100%	0,00	0,05
10	U[8,12]	Melhor Caso	0,60		0,00	0,37		0,00	0,19		0,00	0,04
		Pior Caso	1,05		0,00	5,01		0,00	0,36		0,00	0,07
		Média	-	0%	-	-	0%	-	24,84	100%	-	0,34
15	U[12,18]	Melhor Caso	-		-	-		-	0,61		-	0,19
		Pior Caso	-		-	-		-	44,08		-	0,47
Média Total			6,37		1,15	1,04		3,10	3,00		4,54	0,06

(-) Indica que as instâncias não foram resolvidas em 3600s.
(*) Indica que a média, não inclui as instâncias não resolvidas.

Capítulo 5

Heurísticas Construtivas

No presente capítulo são detalhadas as heurísticas construtivas polinomiais implementadas que resolvem o problema $(F2(P)|CD|C_{max})$ para instâncias de média e grande escala. Primeiramente serão detalhadas as duas heurísticas propostas por Chen e Song [9] que apresentaram melhores resultados (heurísticas JRH e JLPTH). Posteriormente as quatro heurísticas propostas serão apresentadas. De forma que no capítulo seguinte essas heurísticas sejam comparadas. Esta parte do trabalho é uma continuação do trabalho de Araújo e Melo [3], sendo que as heurísticas aqui apresentadas foram adaptadas de forma a diminuir a complexidade computacional das heurísticas inicialmente propostas em [3].

5.1 Heurísticas de Chen e Song

Entre as heurísticas de Chen e Song apenas JRH e JLPTH foram implementadas, uma vez que elas apresentaram resultados superiores às heurísticas dinâmicas DJRH e DJLPTH. As heurísticas estão descritas abaixo, entretanto os leitores podem consultar [9] para maiores detalhes sobre heurísticas e resultados computacionais.

JRH (*Johnson's rule-based ready time heuristic*)

Essa abordagem constrói primeiramente a instância auxiliar, e aplica o algoritmo de *Johnson* nos *jobs* fictícios.

O pseudoalgoritmo de JRH segue abaixo:

1. Para todo $j_{1e_1} \in J_1$, faça:

   $u(j_{1e_1}) = p_{1e_1}$

$v(j_{1e_1}) = \sum_{j_{2e_2} \in J_2 : j_{1e_1} \in S_{e_2}} p_{2e_2} \, |S_{e_2}|$

2. Para todo $j_{1e_1} \in J_1$, faça:

   $E = \{j_{1e_1} | u(j_{1e_1}) < v(j_{1e_1})\}$

   $G = \{j_{1e_1} | u(j_{1e_1}) \geq v(j_{1e_1})\}$

   Ordene os *jobs* de E em ordem não decrescente de $u(j_{1e_1})$ e os *jobs* de G em ordem não crescente de v (j_{1e_1}). Assim, os *jobs* que possuem menor tempo de processamento no primeiro estágio $(u(j_{1e_1}))$ em relação ao segundo são ordenados crescentemente de acordo com os tempos de processamento no primeiro estágio. Os *jobs* restantes são ordenados decrescentemente em função dos tempos de processamento no segundo estágio $(v(j_{1e_1}))$.

   Junte a lista ordenada de G ao final de E. Essa concatenação dá origem a J_1.

3. Para o primeiro estágio, processe os *jobs* segundo a ordem acima J_1, sempre que alguma máquina estiver disponível.

4. Para o segundo estágio, sempre que houver alguma máquina disponível, aloque os *jobs* $j_{2e_2} \in J_2$ segundo seus *ready times*. Ou seja, ordenando crescentemente em função do tempo de completude máximo de seus *jobs* precedentes do primeiro estágio.

No artigo de Chen e Song [9] o passo 1 constroí a lista auxiliar em tempo computacional $O(n_1 n_2)$. O passo 2 finaliza o sequenciamento em $O(n_1 log n_1)$. O passo 3 possui complexidade $O(n_1 m_1)$ e o passo 4 $O(n_1 n_2 m_2 log n_2)$. Dessa maneira esse algoritmo JRH_RT é computado em tempo computacional da ordem de $O(n_1 n_2 m_2 log n_2)$.

JLPTH (*Johnson's rule-based LPT heuristic*)

Nesse caso, segue-se a mesma lógica de JRH, com a diferença de que a ordenação J_2 para o segundo estágio é obtida pela ordenação segundo a regra do *largest processing time* (LPT).

Apenas o passo 4 do pseudoalgoritmo de JRH é alterado: Para o segundo estágio, sempre que houver alguma máquina disponível, processe todos os *jobs* prontos via LPT.

O único passo alterado é o passo 4 que possui complexidade computacional $O(n_1 n_2^2 m_2 log n_2)$, assim o algoritmo JLPTH é resolvido em tempo computacional $O(n_1 n_2^2 m_2 log n_2)$, definido pela ordem do quarto passo.

5.2 Heurísticas Tratadas

Seguindo a lógica das heurísticas supracitadas, uma vez que o objetivo é a comparação entre as heurísticas, consideramos dois pares de heurísticas construtivas propostas por Araújo e Melo [3] que apresentaram melhores resultados.

No trabalho realizado por Araújo e Melo [3] cinco pares de heurísticas foram propostas, lembrando que a diferença entre as heurísticas de cada par está na regra que determina a ordem do sequenciamento durante o segundo estágio: uma ordena em ordem decrescente dos *ready times* e a outra conforme a regra do maior tempo de processsamento (*largest processing time* - LPT). Nesse dissertação iremos tratar os dois pares de heurísticas que apresentaram melhores ressultados, BDm1 e NEHtp2. Essas heurísticas foram embasadas na heurística NEH, a qual foi proposta por Nawaz et al. [19] e é bem conhecida e consagrada por apresentar resultados bem interessantes para os problemas clássicos de *flowshop* permutacional sem grande esforço computacional. Esses dois pares de heurísticas utilizam uma lista de inserção e, a cada passo, avaliam de alguma forma o *makespan*, mesmo sendo ele fictício ou para um dos estágios apenas.

Entretanto de maneira a reduzir a complexidade computacional das heurísticas propostas por Araújo e Melo [3], implementou-se as heurísticas de maneira a simplificar o sequenciamento dos $jobs$, sem haver a necessidade de utilizar a heurística NEH, os mesmos resultados foram obtidos, entretanto a complexidade computacional das heurísticas foi reduzida. Nas subseções seguintes segue o passo a passo de cada uma das heurísticas propostas, sendo BDMP uma atualização da heurísticas BDm1, e FIH a nova heurística proposta a partir de NEHtp2.

As heurísticas propostas utilizam principalmente a heurística construtiva de ordenação denominada LPT (*Longest Processing Time first*). Na heurística LPT os $jobs$ a serem processados são organizadas em uma lista de inserção em ordem decrescente de tempos de processamento, sendo então alocados ao processador menos carregado, essa heurística tem uma complexidade de tempo igual a $O(nlogn)$, dominada pela ordenação inicial dos $jobs$ (n equivalente ao número de $jobs$ a serem sequenciados).

5.2.1 Heurísticas BDMP

Nesta seção, são propostas duas heurísticas similares, sendo que a diferença entre elas está na regra que determina a ordem do sequenciamento durante o segundo estágio.

BDMP se baseia inicialmente na construção de uma ordem de inserção (I_2) para os *jobs* do segundo estágio, obtida pela ordenação decrescente dos tempos de processamentos fictícios T_{e_2}, que consideram os tempos de processamento dos *jobs* do segundo estágio, bem como, os tempos de processamento de seus *jobs* precedentes e o número de máquinas na primeira etapa. Após a construção de I_2, esses *jobs* são sequenciados nas máquinas do segundo estágio sempre que houver disponibilidade, de forma a minimizar o *makespan* local, esse sequenciamento é designado J^2. A partir da sequência J^2 obtida ordena-se os *jobs* do primeiro estágio que pertencem a cada conjunto de precedência no primeiro estágio segundo o LPT, sendo essa a sequencia final para o primeiro estágio. Uma vez definido o sequenciamento dos *jobs* do primeiro estágio deve-se sequenciar novamente os *jobs* do segundo estágio seguindo a regra dos *ready times* na heurística BDMP_RT e segundo a regra LPT na heurística BDMP_LPT.

BDMP_RT

1. Para cada *job* do segundo estágio, crie tempos fictícios T_{e_2}.
   $T_{e_2} = \frac{1}{m_1}(\sum_{j_{1e_1} \in S_{e_2}} p_{1e_1}) + p_{2e_2}$
2. Ordene os *jobs* do segundo estágio $j_{2e_2} \in J_2$ em ordem decrescente de tempos fictícios T_{e_2}. A ordem resultante é chamada ordem de inserção I_2.
3. Sequencie os *jobs* da lista de inserção I_2, na máquina do segundo estágio que estiver disponível primeiramente, de forma a obter o menor *makespan* local. Para o cálculo do *makespan* são utilizados os tempos de processamento dos *jobs* do segundo estágio (p_{2e_2}).
4. Considerando o sequenciamento dos *jobs* no segundo estágio (passo anterior, sequencia J^2), ordenar e sequenciar os seus *jobs* precedentes em ordem decrescente de tempo de processamento.
5. Para o segundo estágio, sempre que houver alguma máquina disponível, aloque os *jobs* $j_{2e_2} \in J_2$ segundo seus *ready times*. Ou seja, ordenando crescentemente em função do tempo de completude máximo de seus *jobs* precedentes do primeiro estágio.

Proposição 5.2.1 *A heurística BDMP_RT possui complexidade computacional de ordem* $O(n_1 n_2)$.

Demonstração O passo 1 possui complexidade computacional de ordem $O(n_1 n_2)$. Para a construção da lista de inserção I_2 tem-se $O(n_2 log n_2)$. O passo 3 sequencia

os *jobs* do segundo estágio nas máquinas, possui complexidade $O(n_2m_2)$. O passo 4 ordena e sequencia os *jobs* do primeiro estágio em tempo computacional $O(n_2logn_2)$ e por fim o passo 5 é da ordem $O(n_2logn_2)$. ∎

BDMP_LPT

Nesse heurística, segue-se a mesma lógica de BDMP_RT, com a diferença de que a ordenação final de J^2 para o segundo estágio é obtida pela ordenação segundo a regra do *largest processing time* (LPT) quando mais *jobs* estão prontos para serem processados em um mesmo momento. Assim, apenas o último passo do pseudoalgoritmo de de BDMP_RT é alterado:

5. Sequencie os *jobs* do segundo estágio em ordem crescente de *release dates* sempre que houver uma máquina disponível. Se ocorrer de mais *jobs* estarem prontos em um mesmo momento, deve-se fazer o processamento segundo a regra LPT.

Proposição 5.2.1 *A heurística BDMP_LPT possui complexidade computacional* $O(n_1n_2)$.

Demonstração A única diferença entre BDMP_RT e BDMP_LPT é o passo 5, entretanto a complexidade computacional para se resolver esse passo também é de ordem $O(n_2logn_2)$, o que não altera a complexidade computacional do algoritmo. ∎

5.2.2 Heurística FIH

Como na seção anterior, um par heurísticas será proposto, sendo que a diferença entre as heurísticas é dada pelo último sequenciamento no segundo estágio, ora via *ready times* ora via LTP.

A heurística FIH utiliza uma lista de inserção I_2, essa lista é composta pelos *jobs* do segundo estágio, ordenados em ordem decrescente de tempos de processamento. Posteriormente esses *jobs* são sequenciados nas máquinas do segundo estágio, sempre que houver disponibilidade. Para a construção deste sequenciamento são criados tempos fictícios de processamento (T_{e_2}) para avaliar a *makespan* local. É importante notar que este tempo de processamento fictício é utilizados para avaliar *makespan* quando esta sequenciando a lista I_2, por isso o *makespan* é chamado local. A partir da solução J^2 obtida, os *jobs* no primeiro estágio são sequenciados considerando o conjunto de precedência (S_2) de acordo com a regra LPT, este será o sequencimento final para os *jobs* pertencentes ao primeiro estágio, designado J^1. Uma vez definida a

solução final no primeiro estágio, J^1, os *jobs* da segunda etapa devem ser novamente sequenciados.

FIH_RT

1. Ordene os *jobs* do segundo estágio $j_{2e_2} \in J_2$ via LPT em uma lista de inserção chamada I_2.
2. Para cada *job* do segundo estágio, crie tempos fictícios T_{e_2}.
   $T_{e_2} = \frac{1}{m_1}(\sum_{j_{1e_1} \in S_{e_2}} p_{1e_1}) + p_{2e_2}$.
3. Sequencie os *jobs* da lista de inserção I_2, na máquina do segundo estágio que estiver disponível primeiramente, de forma a obter o menor *makespan* local. O *makespan* desse sequenciamento é avaliado com os tempos fictícios, calculados no passo anterior e não através dos tempos de processamento dos *jobs*.
4. Considerando o sequenciamento dos *jobs* no segundo estágio (passo anterior, sequencia J^2), ordenar e sequenciar os seus *jobs* precedentes em ordem decrescente de tempo de processamento.
5. Sequencie os *jobs* do segundo estágio em ordem crescente de *release dates* sempre que houver uma máquina disponível.

Proposição 5.2.1 *A heurística FIH_RT possui complexidade computacional de ordem* $O(n_1 n_2)$.

Demonstração O passo 1 ordena os *jobs* em uma lista de inserção segundo a regra LPT, $O(m_2 log m_2)$. O passo 2, calcula os tempos fictícios, complexidade de ordem $O(n_1 n_2)$. O passo 3 sequencia os *jobs* do segundo estágio nas máquinas, possui complexidade $O(n_2 m_2)$ O passo 4 ordena os *jobs* do primeiro estágio em tempo computacional $O(n_2 log n_2)$ e por fim o passo 5 é da ordem $O(n_2 log n_2)$. ∎

FIH_LPT

Essa heurística realiza todo o procedimento realizado em FIH_RT com a diferença de que o *largest processing time* deve ser usado não só para o sequenciamento final do primeiro estágio, mas também para o re-sequenciamento dos *jobs* no segundo estágio. Isto implica na mudança do Passo últimos do algoritmo anterior.

5.Para o segundo estágio, sempre que houver alguma máquina disponível, processe todos os jobs prontos via LPT.

Proposição 5.2.1 *A heurística FIH_LPT possui complexidade computacional de ordem* $O(n_1 n_2)$.

Demonstração A única diferença entre FIH_RT e FIH_LPT é o passo 5, entretanto a complexidade computacional para se resolver esse novo passo também é de ordem $O(n_2 log n_2)$, o que não altera a complexidade computacional do algoritmo.

5.3 Análise de Pior Caso

No capítulo seguinte os resultados são apresentados, como a heurística FIH_LPT apresenta bons resultados para todos os grupos de teste estudados é interessante nessa seção fazer a análise de pior caso da heurística. O pior caso representa um valor mínimo de desempenho do algoritmo. Segundo Fisher [10] o estudo do pior caso estabelece o máximo desvio de otimalidade quando uma heurística específica é aplicada em uma classe de problemas.

Proposição 5.3.1 *Para* $F2(P)|CD|C_{max}$ *tem-se que:*

$$C_{max}(HEU) - C_{max}(OPT) \leq p_{1n}(1 - \frac{1}{m_1}) + p_{2n}(1 - \frac{1}{m_2}) + \overline{P_1} - S' \qquad (5.1)$$

$C_{max}(HEU)$ *denota o makespan para o pior caso e* $C_{max}(OPT)$ *denota o makespan do sequenciamento ótimo (possivelmente desconhecido). As variáveis* p_{1n} *e* p_{2n} *representam os maiores jobs de cada um dos estágios de sequenciamento, já* $\overline{P_1}$, *e* $\overline{P_2}$, *seguem pelas expressões matemáticas abaixo, sendo que* m_i *é o número de máquinas para o estágio* i, $i \in \{1, 2\}$, *e* S' *representa o menor release date* (r_{2e_2}).

$$\overline{P_1} = \sum_{e_1=1}^{n_1} \frac{p_{1e_1}}{m_1}$$

$$\overline{P_2} = \sum_{e_2=1}^{n_2} \frac{p_{2e_2}}{m_2}$$

Demonstração A Proposição 5.3.1, apresenta *gap* absoluto entre o *makespan* calculado em seu pior caso $C_{max}(HEU)$ e o limite inferior $C_{max}(OPT)$ referente ao sequenciamento ótimo, para o problema de *crossdocking* com máquinas paralelas cujo objetivo é minimizar o *makespan* $F2(P)|CD|C_{max}$.

A prova define o pior caso para o problema, ou seja, qual é o maior diferença entre uma solução heurística e o ótimo. Para isso, o limite inferior $C_{max}(OPT)$ é a melhor solução possível, considerando o melhor caso.

O limite superior $C_{max}(HEU)$, é a pior solução possível calculada por uma heurística, considerando o pior caso possível. $C_{max}(HEU)$ é dividido em duas partes, onde, $C_{max}(HEU_1)$ é *makespan* calculado pela heurística para os *jobs* do primeiro estágio (j_{1e_1}), e $C_{max}(HEU_2)$ o *makespan* calculado pela heurística para os *jobs* do segundo estágio (j_{2e_2}). Dessa maneira o $C_{max}(HEU)$ é a soma dos dois *makespan*, representando o pior caso, pois considera que os *jobs* do segundo estágio só iniciam seu processamento após todos os *jobs* do primeiro estágio terem sido precessados. Como o limite superior $C_{max}(HEU)$ é a soma dos *makespans* nos dois estágios, para obter o pior caso é preciso encontrar o pior caso para o primeiro estágio e soma-ló ao pior caso para o segundo estágio. Para isso, considera se que os $n_1 - 1$ *jobs* do primeiro estágio foram sequenciados baseados na sequência provisória de J^2 e que o último *job* a ser sequenciado (j_{1n}) apresente o maior tempo de processamento, p_{1n}. O mesmo é considerado para para os *jobs* da segunda máquina, que são sequenciados em ordem decrescentemente de seus *release dates*, dependem da sequência J^1 e via LPT quando existe empate de *release dates*, dessa forma vamos considerar o pior caso, ou seja, que o último *job* a ser sequenciado j_{2n} apresenta o maior tempo de processamento, p_{2n}. Após essas considerações, tem-se que:

$$C_{max}(OPT) \geq S' + \sum\nolimits_{j_{2e_2} \in J_2} \frac{p_{2e_2}}{m_2} \tag{5.2}$$

sendo que, S' representa o menor *release date* (r_{2e_2}), e $\sum_{j_{2e_2} \in J_2} \frac{p_{2e_2}}{m_2}$ o tempo mínimo necessário para sequenciar todos os *jobs* do segundo estágio. Assim nesse cálculo avaliamos o melhor caso existente, somando o menor *release date* e o melhor sequenciamento possível, é importante lembrar que todos os *jobs* do segundo estágio possuem pelo menos um precedente.

Uma vez definida a melhor solução possível, é preciso definir a pior solução obtida através da heurística:

$$C_{max}(HEU) \leq C_{max}(HEU_1) + C_{max}(HEU_2) \tag{5.3}$$

$$C_{max}(HEU_1) - p_{1n} \leq \sum_{e_1=1}^{n_1-1} \frac{p_{1e_1}}{m_1} \tag{5.4}$$

$$C_{max}(HEU_2) - p_{2n} \leq \sum_{e_2=1}^{n_2-1} \frac{p_{2e_2}}{m_2} \tag{5.5}$$

$$C_{max}(HEU) \leq (\sum_{e_1=1}^{n_1-1} \frac{p_{1e_1}}{m_1} + p_{1n}) + (\sum_{e_2=1}^{n_2-1} \frac{p_{2e_2}}{m_2} + p_{2n}) \tag{5.6}$$

$$C_{max}(HEU) \leq (\sum_{e_1=1}^{n_1} \frac{p_{1e_1}}{m_1} + p_{1n}(1 - \frac{1}{m_1})) + (\sum_{e_2=1}^{n_2} \frac{p_{2e_2}}{m_2} + p_{2n}(1 - \frac{1}{m_2})) \tag{5.7}$$

De forma a simplificar o entendimento faremos:

$$\overline{P_1} = \sum_{e_1=1}^{n_1} \frac{p_{1e_1}}{m_1} \tag{5.8}$$

$$\overline{P_2} = \sum_{e_2=1}^{n_2} \frac{p_{2e_2}}{m_2} \tag{5.9}$$

Retornando à equação (5.7), tem-se o limite superior, é importante ressaltar que todas a variáveis da equação (5.10) apresentam valores conhecidos:

$$C_{max}(HEU) \leq p_{1n}(1 - \frac{1}{m_1}) + p_{2n}(1 - \frac{1}{m_2}) + \overline{P_1} + \overline{P_2} \tag{5.10}$$

De (5.2) tem-se que:

$$\overline{P_2} < C_{max}(OPT) - S' \tag{5.11}$$

Então:

$$C_{max}(HEU) - C_{max}(OPT) \leq p_{1n}(1 - \frac{1}{m_1}) + p_{2n}(1 - \frac{1}{m_2}) + \overline{P_1} - S'$$

∎

Capítulo 6

Experimentos Computacionais - Heurísticas

Os experimentos realizados nessa seção foram executados utilizando-se o processador intel Core 2 Duo @ 2,26 GHz 2,27 GHz, 3,00 GB RAM, em sistema operacional Linux 32 bits, distribuição Ubunto 11.0.4. O software de otimização CPLEX 12.4, bem como, os testes realizados no capítulo 4. As instâncias de teste criadas para avaliar o desempenho das heurísticas foram geradas através do gerador de números pseudo-aleatórios de *Mersenne Twister* a partir de uma semente seguindo o pseudocódigo apresentado em 4.1.

6.1 Lower Bound

Chen e Song [9] propuseram um *lower bound* para o problema que foi aqui utilizado para a comparação da qualidade dos resultados. Esse *lower bound* baseia se no fato de que cada *job* no segundo estágio possui um tempo médio de processamento no primeiro estágio, tempo que, para uma máquina no segundo estágio é ocioso devido às restrições do $flowshop$. Segue pseudo-algoritmo para o cálculo do lower bound:

1. Para cada *job* j_{2e_2} no segundo estágio há um tempo de processamento fictício correspondente no primeiro estágio $p_{e_2}^{(1)}$

$$p_{e_2}^{(1)} = max\{max_{j_{1e1} \in S_{e_2}} p_{1e_1}, \sum_{j_{1e_1} \in S_{e_2}} \frac{p_{1e_i}}{m_1}\}$$

   Os tempos de processamento no segundo estágio $p_{e_2}^{(2)}$ são iguais aos tempos de

processamento originais p_{e_2}

$$p_{e_2}^{(2)} = p_{e_2}$$

2. Ordene os jobs em ordem não-decrescente dos tempos de processamento fictícios do primeiro estágio, e dos q menores tempos de processamento do estágio 2.
3. $p_q^{(i)} = \sum_{e_1=1}^{q} p_{e_i}^{(i)}$ a soma dos q menores tempos de processamento fictícios do primeiro estágio, e dos q menores tempos de processamento do estágio 2.
4. Faça

$$LB = \frac{(p_{m_2}^{(1)}) + (p_{n_2}^{(2)})}{m_2}$$

O ideal seria poder comparar os resultados das heurísticas com o *makespan* ótimo. Porém, para instâncias médias a grandes, o tempo computacional para se chegar ao *makespan* ótimo é inviável. Ainda assim, é necessário ter alguma métrica para avaliar a qualidade dos resultados obtidos pelas heurísticas. Para isso, utilizou-se o GAP em relação ao *lower bound* como medida de qualidade:

$$GAP = \frac{(makespan - lowerbound)}{lowerbound}$$

Este valor tem a importância de ser uma medida que é incontestavelmente maior que a real distância entre o *makespan* e seu valor ótimo.

6.1.1 Metodologia dos experimentos computacionais

Com o intuito de ser fiel aos resultados dos autores supracitados, a mesma metodologia foi adotada para os testes, de tal forma que manteve-se o tamanho das instâncias geradas, que são de médias e grandes escalas, e as medidas de desempenho de cada algoritmo. No artigo dos autores não foram disponibilizadas as instâncias, mas foram indicados o número de *jobs* e número de máquinas para cada estágio aplicados. Sendo assim houve a necessidade de gerar instâncias, que seguiu os passos apresentados na Seção 4.1.

Para cada heurística cinco grupos de máquinas foram analisados, sendo que três deles utilizaram número de máquinas determinístico em cada estágio e dois considerou um número de máquinas aleatório, como segue:

1. Duas máquinas nos dois estágios.
2. Quatro máquinas nos dois estágios.

3. Dez máquinas nos dois estágios.

4. Número de máquinas gerado por uma distribuição uniforme entre dois e quatro ($Unif\,[2,4]$) para cada estágio.

5. Número de máquinas gerado por uma distribuição uniforme entre dois e dez ($Unif\,[2,10]$) para cada estágio.

Dentro de cada grupo, variou-se o número de *jobs* no primeiro estágio, em um intervalo de 10 em 10, entre os valores 20 e 80. Para os *jobs* do segundo estágio, gerou-se um número uniformemente entre 80% e 120% do número de *jobs* do primeiro estágio, para cada instância. Para cada caso 300 instâncias de teste foram solucionadas, totalizando 10500 problemas diferentes resolvidos. Nas instâncias criadas os tempos de processamento dos *jobs* seguiu uma distribuição uniforme entre zero e cem ($Unif[0,100]$) A Tabela 6.1 apresenta um resumo das instâncias geradas.

Tabela 6.1: Variação das instâncias de teste: Resumimos como as instâncias variaram para os testes, a última coluna apresenta os valores de teste que foram considerados.

Parâmetro	Notação	Valores testados
Número de jobs no 1º estágio	n_1	20,30,40,50,60,70,80
Número de jobs no 2º estágio	n_2	[16,24] para n_1 igual a 20
		[24,36] para n_1 igual a 30
		[32,48] para n_1 igual a 40
		[40,60] para n_1 igual a 50
		[48,72] para n_1 igual a 60
		[56,84] para n_1 igual a 70
		[64,96] para n_1 igual a 80
Número de máquinas no estágio i	m_i	2,4,10, $Unif[2,4]$, $Unif[2,10]$
Tempo de processamento	p_{ie_i}	$Unif[10,100]$

Cada subgrupo de 300 instâncias foi considerado como uma amostra, de forma que fosse possível obter os valores médios, piores e melhores casos, assim como no artigo de Chen e Song [9].

6.1.2 Resultados dos experimentos computacionais

Tabela 6.2: Resultados computacionais de média, melhor caso e pior caso, para instâncias de média e grande escala para todos os grupos de máquinas.

m_i	n_1	n_2		$GAP(\%)$					
				JRH	JLPTH	BDMP_RT	BDMP_LPT	FIH_RT	FIH_LPT
2	20	[16,24]	Melhor Caso	24,62	14,15	25,58	19,12	20,95	17,69
			Média	44,92	33,8	47,37	39,86	44,95	35,37
			Pior Caso	82,8	72,24	80,39	70,54	74,33	64,08
	30	[24,36]	Melhor Caso	29,03	17,96	29,42	23,1	29,06	21,31
			Média	45,4	36,07	47,37	39,75	46,18	37,59
			Pior Caso	69,87	53,14	76,02	73,03	72,8	64,72
	40	[32,48]	Melhor Caso	28,9	23,77	32,38	25,81	28,9	24,6
			Média	45,56	37,53	47,12	39,65	46,12	38,67
			Pior Caso	66,17	57,4	80,76	63,91	70,5	59,4
	50	[40,60]	Melhor Caso	28,49	24,76	30,2	27,23	30,46	23,65
			Média	45,31	38,17	46,85	39,5	46,03	38,94
			Pior Caso	67,71	58,48	68,99	60,68	68,31	59,36
	60	[48,72]	Melhor Caso	30,59	26,46	32,57	27,75	30,46	26,65
			Média	45	38,7	46,33	39,53	46,03	38,93
			Pior Caso	63,64	56,47	66,84	57,04	68,31	56,3
	70	[56,84]	Melhor Caso	32,72	27,81	33,36	29,35	32,73	28,02
			Média	45,37	39,69	46,42	40,17	45,57	39,81
			Pior Caso	65,47	59,39	67,79	55,89	62,84	56,35
	80	[64,96]	Melhor Caso	31,43	27,6	33,42	28,7	33,19	28,01
			Média	44,93	39,44	45,73	39,81	45,13	39,58
			Pior Caso	59,34	55,91	61,29	54,93	61,08	55,55
4	20	[16,24]	Melhor Caso	28,21	20,1	27,61	24,72	23,36	17,56
			Média	45,91	37,47	47,69	40,41	43,5	35,89
			Pior Caso	80,03	65,99	76,29	65,49	71,02	59,75
	30	[24,36]	Melhor Caso	28,48	23,9	29,77	25,38	29,16	24,02
			Média	46,12	38,94	47,32	41,17	43,91	37,29
			Pior Caso	67,61	55,8	70,28	60,14	62,78	59
	40	[32,48]	Melhor Caso	31,76	26,44	33,6	28,08	30,7	22
			Média	45,68	39,34	46,34	41,13	43,64	37,64
			Pior Caso	64,61	56,2	74,67	67,54	65,34	54,92
	50	[40,60]	Melhor Caso	31,25	26,21	28,64	24,75	28,28	21,9
			Média	44,82	39,51	45,49	40,76	43,39	37,66
			Pior Caso	62,01	55,25	68,12	57,62	64,32	57,31
	60	[48,72]	Melhor Caso	30,38	27,63	32,77	27,39	29,51	26,52
			Média	44,55	39,67	44,82	40,68	42,7	37,7
			Pior Caso	61,29	53,71	59,94	54,01	56,83	51,43
	70	[56,84]	Melhor Caso	32,97	28,87	29,88	29,16	30,76	26,71
			Média	44,55	40,32	44,86	40,73	42,75	38,21
			Pior Caso	59,63	56,06	62,91	55,07	60,63	52,69
	80	[64,96]	Melhor Caso	32,5	28,41	31,51	30,54	32,02	27,63
			Média	44,04	40,01	44,21	40,63	42,67	38,33
			Pior Caso	58,68	53,93	60,89	54,9	55,26	50,97
10	20	[16,24]	Melhor Caso	19,2	11,92	19,28	7,7	17,83	5,69
			Média	39,48	28,94	39,54	28,98	38,55	28,23
			Pior Caso	64,81	62,94	65,38	65,38	63,69	61,03
	30	[24,36]	Melhor Caso	30,12	20,99	25,77	19,99	29,41	18,53

			Média	48,8	35,88	47,91	36,02	46,26	43,65
			Pior Caso	69,11	56,18	69,42	53,34	69,04	53,51
	40	[32,48]	Melhor Caso	35,75	26	34,66	25,73	31,02	27,2
			Média	54,01	42,78	52,59	41,57	51,37	40,87
			Pior Caso	76,35	59,58	80,9	64,76	70,88	57,14
	50	[40,60]	Melhor Caso	40,08	33,66	37,55	29,8	35,33	27,36
			Média	54,63	45	53,48	43,87	52,28	42,83
			Pior Caso	75,77	57,25	77,83	57,81	68,77	57,84
	60	[48,72]	Melhor Caso	37,36	33,14	36,44	29,84	36,41	30,42
			Média	52,68	44,23	51,71	43,08	50,22	41,97
			Pior Caso	70,2	59,27	71,27	56,73	66,39	57,24
	70	[56,84]	Melhor Caso	34,21	32,02	36,55	30,41	36,05	29,75
			Média	50,89	43,86	49,69	42,47	48,53	41,24
			Pior Caso	66,09	55,91	66,52	54,78	67,19	52,51
	80	[64,96]	Melhor Caso	36,49	32,37	37,29	32,06	34,62	30,17
			Média	49,57	43,21	48,31	41,89	47,38	40,82
			Pior Caso	68,81	54,83	64,65	53,7	64,27	51,81
Unif[2,4]	20	[16,24]	Melhor Caso	14,92	11,46	13,4	12,86	11,35	9,02
			Média	45,43	36,61	48,03	41,66	43,88	36,48
			Pior Caso	97,48	84,26	100,94	91,02	89,67	86,91
	30	[24,36]	Melhor Caso	19,02	14,09	18,6	16,87	13,88	10,14
			Média	46,4	38,54	48,1	42,16	45,33	38,14
			Pior Caso	93,49	81,63	95,8	88,71	93,49	85,47
	40	[32,48]	Melhor Caso	15,97	10,95	19,37	14,46	15,72	10,61
			Média	45,42	38,69	46,88	41,32	44,62	38,18
			Pior Caso	93,04	79,2	91,16	88,47	88,88	79,1
	50	[40,60]	Melhor Caso	16,44	14,38	17,05	12,44	15,49	12,64
			Média	45,54	39,61	46,74	41,54	44,73	38,69
			Pior Caso	89,03	77,06	86,12	81,32	85,48	76,62
	60	[48,72]	Melhor Caso	18,03	14,52	18,34	13,21	17,19	11,83
			Média	45,01	39,51	45,81	40,76	44,34	38,74
			Pior Caso	93,71	83,24	94,2	90,25	93,07	81,55
	70	[56,84]	Melhor Caso	20,66	18,57	21,28	16,69	18,92	16,39
			Média	45,01	40,08	45,55	40,74	44,52	39,3
			Pior Caso	85,8	74,35	84,43	81,36	84,17	77,17
	80	[64,96]	Melhor Caso	19,16	18,08	20,11	16,54	18,8	15,48
			Média	44,88	40,35	45,44	40,8	44,39	39,57
			Pior Caso	81,64	78,32	82,42	79,31	82,1	77,62
Unif[2,10]	20	[16,24]	Melhor Caso	4,94	3,28	3,91	1,77	2,23	2,11
			Média	45,64	37,88	47,05	38,82	43,63	35,94
			Pior Caso	110,9	118,45	115,88	114,41	110,58	103,55
	30	[24,36]	Melhor Caso	7,38	6,08	5,92	5,49	4,95	4,09
			Média	49,69	41,18	50,41	42,55	47,46	40,02
			Pior Caso	105,09	93,76	110,87	101,37	106,17	97,89
	40	[32,48]	Melhor Caso	9,35	7,39	8,57	6,45	6,99	4,5
			Média	49,62	42,44	50,37	43,36	47,68	40,85
			Pior Caso	98,76	94,39	102,75	98,64	99,37	96,09
	50	[40,60]	Melhor Caso	8,99	8,41	7,99	6,66	5,41	4,61
			Média	49,42	43,29	49,46	43,75	47,52	41,28
			Pior Caso	105,05	99,01	106,36	102,63	104,81	100,21
	60	[48,72]	Melhor Caso	9,35	8,73	8,5	7,92	7,52	6,36
			Média	48,26	42,73	48,32	43,11	46,54	41,03
			Pior Caso	97,87	90,49	102,86	99,04	103,74	98,85
	70	[56,84]	Melhor Caso	10,84	10	9,77	7,25	8,66	6,63

		Média	47,71	42,79	47,51	42,66	46,19	40,85
		Pior Caso	99,12	91,97	101,52	96,74	99,12	93,43
80	[64,96]	Melhor Caso	10,22	10,18	8,28	7,38	9,31	5,94
		Média	47,19	42,68	47,08	42,67	45,87	41,1
		Pior Caso	99,46	88,28	98,52	93,68	96,88	92,11

2 máquinas em cada estágio

Para instâncias com duas máquinas em cada estágio, variando-se o número total de *jobs*, os testes apontam a heurística JLPTH como a melhor em todos os casos, na média. Entretanto a medida que o número de *jobs* aumenta a heurística FIH_LPT passa a apresentar resultados próximos aos encontrados por JLPTH, como mostra o gráfico 6.1

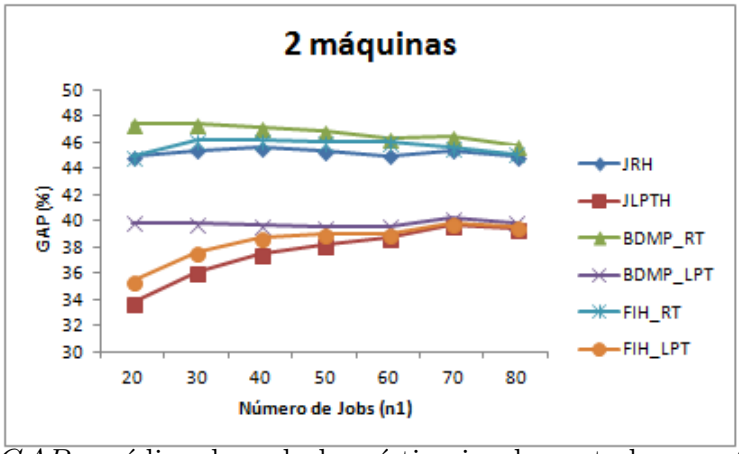

Figura 6.1: *GAPs* médios de cada heurística implementada para 2 máquinas em cada estágio.

Além do valor médio dos *GAPs* de cada algoritmo, é interessante também avaliar o melhor e o pior caso. O pior caso é tão importante quanto o resultado médio das heurísticas, por representar um valor mínimo de desempenho do algoritmo. Devido à importância do pior caso na seção 5.3.1 uma demonstração analítica de pior caso do problema geral é realizada. Apesar de JLPTH ser melhor do que todas as outras na média, JLPTH nem sempre tem o menor valor para o melhor caso e para o pior caso, mas os possui em maior número, o que pode ser verificado na Tabela 6.2.

4 máquinas em cada estágio

O teste realizado com quatro máquinas em cada estágio aponta como melhor heurística a FIH_LPT, na média. O gráfico da Figura 6.2 resume os resultados.

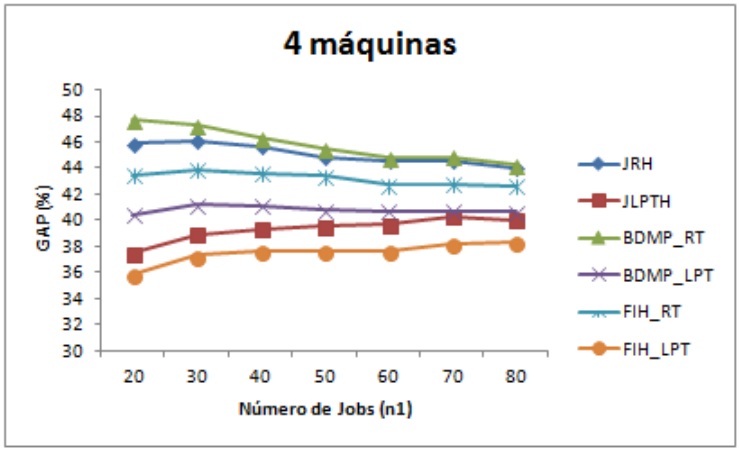

Figura 6.2: *GAPs* médios de cada heurística implementada para 4 máquinas em cada estágio.

A partir da Tabela 6.2 para 4 máquinas é perceptível a superioridade dos resultados alcançados pela heurística proposta FIH_LPT. Quando 2 máquinas foram consideradas, houve uma maior variabilidades dos melhores resultados, alguns foram alcançados pela heurística FIH_LPT e alguns por BDMP_LPT, já para duas máquinas a heurística BDMP_LPT não apresentou na média resultados superiores com relação as demais heurísticas.

Para as instâncias de teste analisadas considerando quatro máquinas o *GAP* médio apresentado pela heurística FIH_LPT foi superior em todos os grupos, apesar do melhor caso na média e do pior caso na média não pertencerem à essa heurística em todos os casos, na maior parte deles pertenceu, esses fatos confirmam os bons resultados obtidos pela heurística proposta FIH_LPT.

10 máquinas em cada estágio

O teste realizado com 10 máquinas em cada estágio visa o comportamento das heurísticas à luz de um sequenciamento mais esparso, onde a proporção do número de *jobs* sobre o número de máquinas é bem menor do que os casos anteriores. O gráfico da Figura 6.3 e a Tabela 6.2 já apresentada ilustram os resultados.

O gráfico da Figura 6.3 nos mostra que JLPTH apresenta resultados piores que os apresentados nos grupos anteriores, ao se comparar seu *GAP* médio com o das outras heurísticas. Nesse ponto que dez máquinas são consideradas a heurística FIH_LPT, apresentou-se superior as demais da mesma maneira que aconteceu para os testes com 4 máquinas em cada estágio, entretando no grupo de 10 máquinas a heurística BDMP_LPT também apresenta resultados na média superiores ao produzidos por JLPTH.

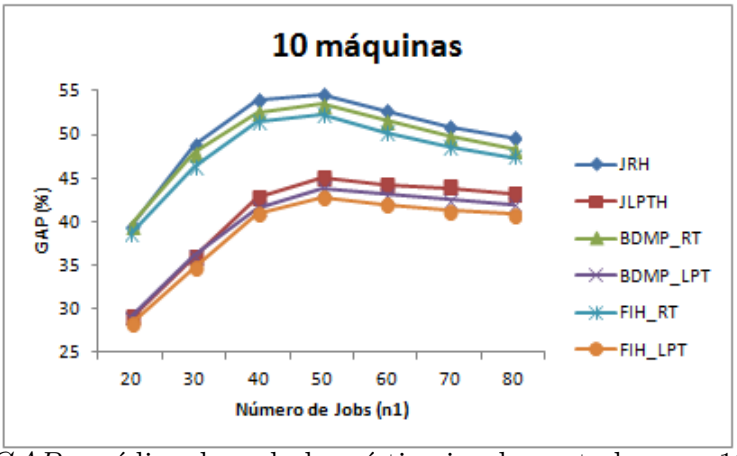

Figura 6.3: *GAPs* médios de cada heurística implementada para 10 máquinas em cada estágio.

Valores uniformes entre dois e quatro para número de máquinas em cada estágio

Com a finalidade de permitir maior flexibilidade nos testes, Chen e Song [9] propuseram também abordagens com números diferentes de máquinas em cada estágio, de modo que os valores de números de máquinas fossem gerados através de uma distribuição uniforme. Analisando primeiramente o grupo com distribuição uniforme entre $Unif[24]$. De acordo com o gráfico da Figura 6.4 e com a Tabela 6.2, a heurística FIH_LPT continua sendo a melhor, em média.

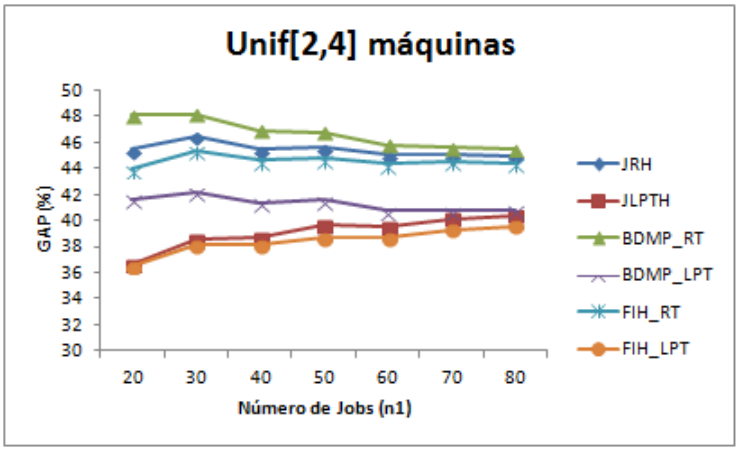

Figura 6.4: *GAPs* médios de cada heurística implementada para Unif[2,4] máquinas em cada estágio.

Valores uniformes entre dois e dez para número de máquinas em cada estágio

Apesar desse grupo possuir alto grau de liberdade, esta abordagem reforça o desempenho de todas as heurísticas perante cenários quaisquer, ou seja, com instâncias de vários tipos distintos, de forma que isto traga um resultado médio para todas elas, sem a dependência do tipo de instância.

A aleatoriedade cedida ao número de máquinas em cada estágio nos mostra que, na média, o *GAP* mínimo das heurísticas pertence a FIH_LPT, apesar do melhor *GAP* dos piores casos das heurísticas pertence à JLPTH na maior parte das vezes. O que pode ser verificado pelo gráfico da Figura 6.5 e pela Tabela 6.2.

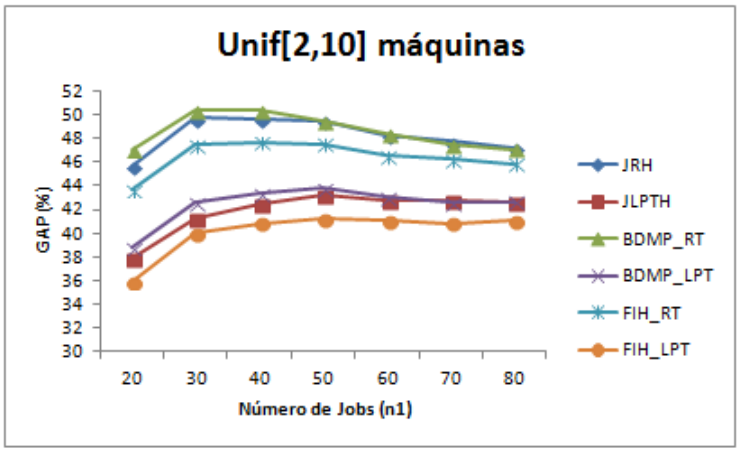

Figura 6.5: *GAPs* médios de cada heurística implementada para Unif[2,10] máquinas em cada estágio.

Capítulo 7

Conclusões e perspectivas

Essa dissertação conseguiu resolver com qualidade um problema real que cada dia tem maior importância no mundo globalizado, o problema de sequenciamento de caminhões em centros de *crossdocking* com múltiplas docas, tanto para centros de pequeno porte quanto para os de médio e grande porte.

Com a finalidade de resolver os problemas de pequeno porte apresentamos um novo modelo de Programação Matemática que empregou a indexação no tempo. Esse modelo resolve na otimalidade problemas de até 10 *jobs* em tempo computacional viável. O modelo foi testado através de suas relaxações lineares, e devido ao fato de utilizar a formulação indexada no tempo produz limites de Relaxação Linear fortes, próximos dos valores ótimos, entretanto, os tempos computacionais necessários para se resolver o problema com o novo modelo continuam sendo altos. Isto ocorre pelo elevado número de restrições e variáveis envolvidos no modelo introduzido. Ainda com relação ao modelo matemático proposto provamos sua dominância sobre o modelo presente na literatura proposto por Chen e Song [9] para resolver o problema estudado. Dessa maneira o modelo matemático proposto apresentou-se de qualidade, e de fácil implementação que resolve de maneira eficiente problemas de pequena escala.

Devido ao fato dos centros de *crossdocking* reais apresentarem um número maior de *jobs* que os resolvidos pelo modelo matemático, apresentamos quatro novas heurísticas. Essas heurísticas foram estudadas inicialmente por Araújo e Melo [3] entretanto, devido a grande importância de se estudar grandes centros tornou viável a implementação e continuação do estudo dos autores a fim de comparar os resultados produzidos aos obtidos pelas por Chen e Song [9], com o objetivo de melhorar os resultados da literatura, para isso centros de até 10 docas e 80 *jobs* foram analisados. Os resultados obtidos foram interessantes, e as instâncias testadas resolvidas em tempo computacional viável. Comparando a heurística pro-

posta às heurísticas propostas em [9], a JLPTH apresentou resultados superiores para os testes que consideraram duas máquinas, entretando quando o número de máquinas aumentou a heurística proposta FIH_LPT se mostrou superior às presentes na literatura, conseguindo resultados de *Gaps* menores que os alcançados em [9]. Ainda para as heurísticas um limite superior foi provado.

Diante desses fatos, bons resultados foram obtidos e os objetivos inicialmente propostos por essa dissertação foram alcançados. Através desse estudo melhorias foram alcançadas na resolução do problema de sequenciamento de caminhões em CCD com múltipas docas, tanto para pequenos centros (através da proposta de um modelo com limites de relaxação mais fortes) quanto para grandes CCD (através da implementação de novas heurísticas construtivas polinomiais). Para os dois casos analisados foi possível alcançar resultados superiores aos produzidos por Chen e Song em [9], ampliando e melhorando o referêncial bibliográfico atual do tema tratado.

Como propostas para trabalhos futuros pretendemos investigar a melhoria nos resultados através da implementação da metaheurística *iterated greedy* utilizando a heurística construtiva proposta FIH_LTP para a construção da solução inicial. Posteriormente o objetivo é avançar esse trabalho com tratamento de processos estocásticos, utilizando o conceito de *simheuristics*, que combina a utilização de simulação e heurísticas.

Referências Bibliográficas

[1] Cross docking: How to use the ean - ucc standards.

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

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

[4] Belle, J. V., Valckenaers, P., and Cattrysse, D. Cross-docking: State of the art. *Omega: The International Journal of Management Science 40* (2012), 827–846.

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

[6] Boysen, N., and Fliedner, M. Cross dock scheduling: Classification, literature review and research agenda. *Omega: The International Journal of Management Science 38* (2010), 413–422.

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

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

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

[10] Fisher, M. L. Worst-case analysis of heuristic algorithms. *Omega Management Science 26* (1980), 1–17.

[11] Garey, M. R., and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co. New York, 1979.

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

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

[14] Lee, C. Y., and Vairaktarakis, G. L. Minimizing makespan in hybrid flowshops. *Operational Research Letter. 16* (1994), 149–158.

[15] Lima, M. F. 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, 2014.

[16] Lira, E. G. Um algoritmo iterated greedy para o problema de sequenciamento de caminhões em centros de cross-docking. Master's thesis, Universidade Federal de Minas Gerais, 2013.

[17] McWilliams, D. L., Stanfield, P. M., and Geiger, C. D. The parcel hub scheduling problem: A simulation-based solution approach. *Computers & Industrial Engineering 49* (2005), 393–412.

[18] Miao, Z., Lim, A., and Ma, H. Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research . 192* (2009), 105–115.

[19] Nawaz, M., Jr, E. E. E., and Ham, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega: The International Journal of Management Science. 11* (1983), 91–95.

[20] Simchi-Levi, D., Kaminsky, P., and Simchi-Levi, E. *Managing the Supply Chain: The Definitive Guide for the Business Professional.* McGraw-Hill, 2003.

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

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

[23] Yu, W., and Egbelu, P. J. Scheduling of inbound and outbound trucks in cross docking systems with temporary storage. *European Journal of Operational Research 184* (2008), 377–396.