Universidade Federal de Minas Gerais

Programa de Pós-Graduação em Engenharia de Produção

Departamento de Engenharia de Produção / Escola de Engenharia

O Problema de Sequenciamento de Caminhões em um Centro de *Crossdocking* com Duas Máquinas

Gabriela Braga Fonseca

Orientador: Prof. Dr. Martín Gómez Ravetti

Belo Horizonte
Novembro, 2015

Universidade Federal de Minas Gerais

Programa de Pós-Graduação em Engenharia de Produção

Departamento de Engenharia de Produção / Escola de Engenharia

O Problema de Sequenciamento de Caminhões em um Centro de *Crossdocking* com Duas Máquinas

Gabriela Braga Fonseca

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 Gómez Ravetti

Belo Horizonte
Novembro, 2015

Resumo

Problemas de sequenciamento da produção lidam com a atribuição de recursos escassos para tarefas ao longo do tempo. Trata-se de um processo de tomada de decisão com o objetivo de otimizar um ou mais objetivos, vital para a competitividade das empresas. O presente trabalho visa desenvolver formas eficientes para resolver o problema de sequenciamento de caminhões em um centro de *crossdocking*, denotado por $F2|CD|C_{max}$, e formulado como um problema de sequenciamento do tipo *flowshop* com duas máquinas, com restrições de *crossdocking*, no qual a função objetivo busca minimizar o *makespan* (C_{max}). Para isso, um modelo de programação linear inteira com formulação baseada em indexação no tempo é considerado. Para validar e avaliar as soluções foram realizados testes com 500 instâncias. Implementamos a Relaxação Lagrangeana através do Algoritmo do Volume, com o objetivo de obter boas soluções em tempo computacionalmente eficiente. Além disso quatro heurísticas polinomiais foram propostas e analisadas, de forma a encontrar um limite superior para o problema. Os resultados obtidos mostraram a eficiência da relaxação lagrangeana e das heurísticas implementadas.

Palavras-Chave: Programação Linear Inteira, Sequenciamento, *Flowshop*, *Crossdocking*, Relaxação Lagrangeana.

Abstract

Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives. This work aims to develop effective ways to solve the truck scheduling problem at a crossdocking facility, denoted by $F2|CD|C_{max}$ and formulated as a scheduling problem of flowshop type with two-machines with crossdocking constraints, in which the objective function seeks to minimize the makespan (C_{max}). For this, an integer linear programming model based in a time-indexed formulation is considered. To confirm and evaluate the solutions tests with 500 instances were performed. We implemented the Lagrangian Relaxation by Volume Algorithm, in order to obtain good solutions in time computationally efficient. Furthermore four polynomial heuristics have been proposed and analyzed in order to find an upper bound for problem. The results demonstrate the efficiency of lagrangean relaxation and implemented heuristics.

KeyWords: Integer Linear Programming, Flowshop, Crossdocking, Lagrangean Relaxation.

Agradecimentos

Agradeço primeiramente à Deus por permitir a realização deste trabalho.
À minha família, minha imensa gratidão por sempre me apoiarem nos meus estudos, não medindo esforços para isso.
Ao professor Doutor Martín Gómez Ravetti pela orientação, que guia esse trabalho com infinita competência.
Ao professor Doutor Thiago Henrique Nogueira, agradeço por estar sempre disposto a me ajudar. Obrigado pela paciência, dicas e orientações que me deixavam mais confiante quando as coisas pareciam não caminhar.
Aos professores, funcionários e colegas do Departamento de Engenharia de Produção.
À CAPES pelo apoio financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Diante da grande valorização por diferenciais competitivos que agreguem valor para o reconhecimento global de uma empresa, é imprescindível que se valorize a agilidade, a excelência e a satisfação dos clientes, bem como a redução dos custos das empresas. Em meio a este cenário, buscam-se alternativas para que se consiga alcançar tais melhorias nos processos logísticos, sem que a qualidade seja afetada. Uma dessas alternativas pode ser o *crossdocking*.

Crossdocking é uma abordagem que elimina ou reduz significativamente duas funções dispendiosas dos centros tradicionais de distribuição que são a estocagem e coleta dos produtos. Os Centros de *Crossdocking* (CCD) são pontos intermediários na rede de suprimentos em que se realiza o transbordo de cargas, sem intenção de estocagem, recebendo caminhões com cargas completas de diversos pontos de fornecimento. Dentro dos CCD, as cargas são retiradas, separadas, combinadas e recarregadas em caminhões de saída, de acordo com os pedidos específicos dos clientes.

Este trabalho trata o problema de sequenciamento de caminhões em um centro de *crossdocking*, denotado por $F2|CD|C_{max}$, e interpretado como um problema de *flowshop* com duas máquinas, com restrições de *crossdocking*, no qual a função objetivo busca minimizar o *makespan* (C_{max}). O problema consiste em determinar uma sequência de descarregamento dos caminhões que chegam aos CCD e uma sequência de carregamento dos caminhões que saem dos centros de forma a minimizar o *makespan*. O modelo aqui considerado foi proposto inicialmente por Chen e Lee [16] e posteriormente estudado por Lima [27]. Essa dissertação baseia-se nos trabalhos desenvolvidos por esses autores, com o intuito de ampliar e melhorar os resultados presentes na literatura.

Dessa forma, primeiramente estudamos o modelo de programação linear inteira com formulação baseada em indexação no tempo, posteriormente elaboramos a relaxação

lagrangeana para o problema, implementamos o algoritmo do volume para resolvê-la, e demonstramos que os subproblemas podem ser resolvidos em tempo polinomial. A fim de obter um limiter superior para o problema desenvolvemos quatro heurísticas polinomiais.

1.1 Objetivos

O objetivo principal desta dissertação de mestrado é estudar formas eficientes para resolver o problema de sequenciamento de caminhões em um centro de *crossdocking*. Para isso, um modelo de programação inteira com formulação baseada em indexação no tempo é estudado. Foram implementados o método de Relaxação Lagrangeana e quatro heurísticas construtivas polinomiais, visando obter limites próximos da solução ótima do problema.

1.2 Organização do Texto

No capítulo 1 é introduzido o tema da pesquisa e apresentado os objetivos do trabalho. O restante deste trabalho é organizado como segue. No capítulo 2 é feita a revisão bibliográfica, definindo o que são Centros de *Crossdocking* e apresentando trabalhos relacionados ao tema. No capítulo 3 é apresentada a formulação matemática proposta para o problema de sequenciamento. No capítulo 4 é apresentada a relaxação lagrangeana do problema, o algoritmo do volume e as quatro heurísticas construtivas polinomiais. Os resultados dos experimentos computacionais são apresentados no capítulo 5. O capítulo 6 apresenta algumas conclusões, bem como as perspectivas para trabalhos futuros.

Capítulo 2

Crossdocking

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). Muitas vezes, as redes de distribuição de mercadorias são ineficientes e geram custos desnecessários por possuirem estruturas baseadas em um modelo histórico ou em experiências passadas.

Um modelo comum 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 uma encomenda de cada fornecedor que possui, como apresentado na Figura 2.1.

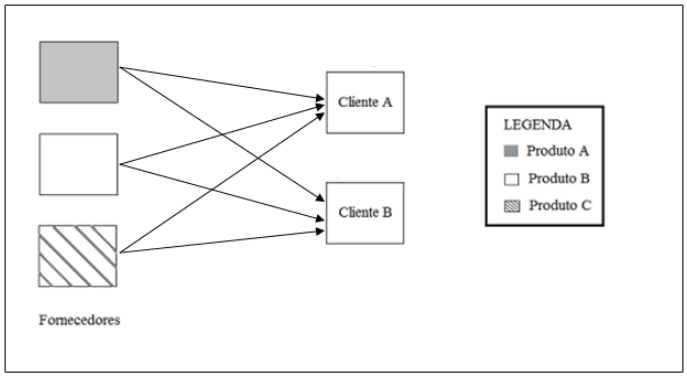

Figura 2.1: Modelo comum de distribuição.

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 frete ou então, caminhões grandes entregam com carga parcialmente cheia. Em ambos os casos, o receptor deve possuir uma capacidade 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, Centros de *Crossdocking* (CCD) funcionam com um estoque limitado ou, se possível, nulo.

O sistema *Crossdocking* apresenta um grande potencial para controlar os custos de logística e distribuição e para manter o nível de serviço aos clientes, já que busca eliminar ou reduzir o estoque não produtivo na cadeia de suprimentos e, junto à ele, eliminar também os custos, o tempo e o trabalho necessário para o seu gerenciamento. O *Crossdocking* possui então, uma capacidade de reduzir os custos de forma estratégica, pois essa redução não afeta o seu nível de serviço, ou seja, os produtos tornam-se mais disponíveis aos clientes e sua entrega mais rápida.

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 completas 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 destino específico. O sequenciamento é realizado tanto no descarregamento quanto no carregamento dos caminhões. 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:

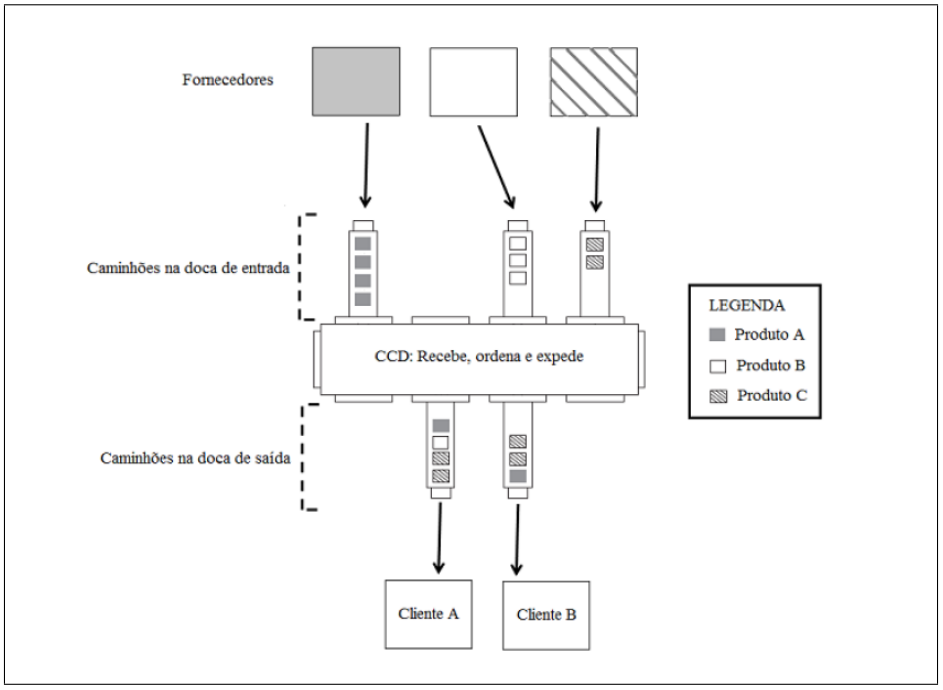

Figura 2.2: Rede de distribuição com *Crossdocking*. - Fonte: [28]

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 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 sua capacidade.

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 [12]. 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. [35] 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 devido ao fato de focar nas necessidades dos clientes de modo que o produto esteja onde e quando o consumidor quer, com custos compatíveis. Para isto, foi fundamental a utilização do centro de *crossdocking*.

2.2 Trabalhos Relacionados

O aumento no uso de técnicas de *crossdocking* e a consequente maior exposição do tema tem motivado diversos autores a pesquisar novas maneiras de melhorar as operações associadas a técnica. Existem diferentes abordagens a respeito dos CCD com enfoques nas diversas etapas do processo logístico.

Boysen e Fliedner [12] e Belle et al. [9] focam na revisão dos trabalhos presentes na literatura que possuem o tema *crossdocking* como foco principal. A diferença entre as duas revisões está no fato de Boysen e Fliedner [12] considerarem não apenas as características de um sistema *crossdocking* como Belle et al. [9] fizeram, mas também incorporaram características de modelos matemáticos de sequenciamento.

Centros de *crossdocking*, assim como unidades de produção, enfrentam problemas envolvendo tomada de decisões desde o nível estratégico até o nível operacional. Segundo Boysen e Fliedner [12] esses problemas podem ser alocados de acordo com a seguinte classificação, ordenada do nível estratégico ao operacional:

- Localização dos CCD;
- *Layout*;

- Roteamento de veículos;
- Atribuição de docas;
- Sequenciamento de caminhões;
- Sequenciamento interno de recursos.

O presente trabalho possui foco principal no problema de sequenciamento de caminhões. A seguir serão descritos os trabalhos relacionados aos tópicos acima, baseando-se nos trabalhos dos autores citados.

A decisão de onde instalar um ou mais CCD faz parte de um conjunto de decisões estratégicas que compõem a elaboração da cadeia de abastecimento. O custo de instalação de um centro de *crossdocking* é uma função composta pelo custo fixo de instalação e um custo variável por unidade carregada. Dessa forma, trabalhos relacionados aos problemas de localização de facilidades e que abordam a forma pela qual os bens devem fluir pela cadeia são relevantes nesse contexto.

Boysen e Fliedner [12] citam trabalhos de Campbell [15] e Klose e Drexl [22], que focam no problema de localização de facilidades, investigando a introdução de pontos intermediários em redes de distribuição. Dentre os autores que incorporam a questão de distribuição dos bens, além da localização, estão Gümüs e BookBinder [10], que modelam a distribuição de bens dos fornecedores aos clientes via *crossdocking*, porém nesse trabalho carregamentos diretos são permitidos e diferentes tipos de produtos são considerados (*multicommodity*).

Gümüs e BookBinder [10] apresentam modelos para redução do custo total em três tipos de redes, cuja diferenciação está na variação da quantidade de fornecedores, produtos e clientes trabalhados. Belle et al. [9], em [10] propõe um programa inteiro misto para solucionar o problema e segundo os autores o número ótimo de CCD é uma função crescente da razão entre o custo fixo do caminhão sobre o custo fixo de instalação de um terminal.

Sung e Song [38] consideraram o mesmo problema, porém carregamentos diretos não são permitidos. Dada a demanda, o objetivo do modelo de programação inteira elaborado é calcular quantos CCD instalar e quantos veículos devem ser usados em cada centro de forma a minimizar os custos associados. O problema é tido como NP-difícil e por isso o algoritmo de busca tabu foi utilizado para auxiliar na busca de melhores resultados. Jayaraman e Ross [21] utilizaram o mesmo algoritmo de busca tabu e também *simulated annealing* para construir heurísticas que auxiliam o modelo com formulação inteira a solucionar instâncias compostas por um fornecedor, várias famílias de produtos, múltiplos centros de *crossdocking* e clientes que demandam vários tipos de produtos. Os resultados são bem próximos à solução ótima.

Outra abordagem está relacionada ao *layout*, que é a forma e dimensão de um centro de *crossdocking*, assim como a forma e a dimensão de seus componentes internos [9]. Bartholdi e Gue [7] buscam determinar qual o melhor formato para um centro de *crossdocking*. A conclusão é que o melhor formato depende do tamanho da instalação e do padrão de fluxo de materiais.

Muitas vezes as cargas destinadas a um centro de *crossdocking* necessitam ser recolhidas e entregues em diferentes pontos e, depois de passar pelo CCD, ser entregue em diferentes localizações. Assim, ambas as etapas podem ser vistas como problema de roteamento de veículos, que associado ao tema *crossdocking*, é objeto de trabalho de alguns autores.

Lee et al. [23] foram os primeiros a dar ênfase nessa área que aborda roteamento de veículos e *crossdocking* simultaneamente. O foco dos autores é identificar uma sequência de roteamento ótima para coleta e entrega, que minimize a soma dos custos de transporte e os custos fixos dos veículos. Os autores apresentam um modelo de programação inteira para o problema, o qual não é capaz de solucioná-lo de forma satisfatória. Assim, um algoritmo de busca tabu é proposto para solucionar o problema.

Wen et al. [18] estudaram o problema e propõem um modelo cujo objetivo é minimizar a distância total percorrida pelos caminhões com restrições de janela de tempo. Cada veículo parte do CCD e deve coletar cargas em fornecedores específicos, retornar ao centro de *crossdocking* e, após o transbordo de cargas, executar uma rota de entrega de produtos a clientes específicos, respeitando a janela de tempo de cada um dos pontos de coleta ou entrega. É proposto pelos autores um modelo de programação linear inteira mista e uma heurística baseada em busca tabu, que pode resolver problemas com até 200 nós alcançando resultados a 5% do ótimo em um curto intervalo de tempo.

Boas alocações de docas podem aumentar a produtividade dos centros de *crossdocking* e reduzir os custos de manuseio [9]. Belle et al. [9] 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. Dessa forma, o problema de alocação de docas procura definir a alocação ótima dos caminhões às docas, 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. Assume-se que o número mínimo de docas seja o mesmo número de caminhões, assim cada caminhão será alocado a diferentes docas 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, foco deste trabalho.

De acordo com Boysen e Fliedner [12], a alocação de docas pode ser executada em um horizonte de curto a médio prazo. Belle et al. [9] afirmam que muitos artigos trabalham com essa abordagem sob um horizonte de 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, entretanto essa prática pode apresentar um alto custo devido a 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 [39, 40] foram os primeiros a estudar o problema de atribuição de docas a médio prazo. Em [39] 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 [40] 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 [13] consideram um centro de *crossdocking*, com o mesmo número de docas de entrada e saída, na qual o transbordo de cargas é feito à noite, podendo a atribuição mudar de uma noite para outra. Os autores propõem inicialmente uma formulação baseada no problema de atribuição quadrática retilínea para minimizar o esforço de manipulação de materiais. Essa é tida como ineficiente para solucionar instâncias de larga escala, considerando que é uma decisão de curto prazo e que deve ser tomada a cada dia. Assim, é utilizada uma heurística baseada em *simulated annealing*, que obtém a 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.

Em Gue [20] existe uma mudança de foco do trabalho, sua análise baseou-se 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 de *crossdocking*. 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.

Para os casos reais, geralmente as docas de entrada e saída não são previamente fixadas, de modo que a atribuição das docas é parte do problema de sequenciamento de caminhões.

O problema de sequenciamento de caminhões se concentra sobre a sucessão dos caminhões de chegada e saída nas docas de um centro de *crossdocking*, ou seja, onde e quando os veículos devem ser processados. Boysen e Fliedner [12] apresentam uma classificação com alguns trabalhos que abordaram o tema de sequenciamento de caminhões. 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. Essas características foram representadas pela notação $\alpha|\beta|\gamma$, relacionando-se com a forma de trabalhar

das docas, as características operacionais e os objetivos que guiaram a otimização, respectivamente. A revisão apresentada por Belle et al. [9] também abordam a classificação citada anteriormente, porém o foco principal está na classificação por características dos CCD.

McWilliams et al. [29] apresentaram o primeiro trabalho focado em sequenciamento de caminhões com um horizonte curto de planejamento. Os autores consideram um centro de *crossdocking* com docas que podem ser de entrada ou saída, o tempo de transporte varia de acordo com as docas atribuídas e o objetivo é minimizar o *makespan*. Para solucionar o problema, é utilizado um algoritmo genético combinado com simulação. Em termos de notação em tuplas, o modelo é determinado como $\mathrm{E} \mid p_j = p, no - wait, t_{i0} \mid C_{max}$. Nesse caso 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 é apresentada por Boysen [11]. Seu trabalho aborda a alocação de docas a caminhões que chegam ao centro de *crossdocking* 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 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 centro de *crossdocking*, no qual cada doca é exclusiva para operações de carregamento ou descarregamento e não é possível armazenar os produtos, o que é comum para bens que necessitam de refrigeração. Também são apresentas heurísticas baseadas em *simulated annealing* para obter limites próximos ao valor ótimo para instâncias reais, representadas por mais de 25 caminhões de chegada. Segundo Belle et al. [9], os modelos trabalhados em [11] seguem as seguintes notações $E\big|p_j = p, no - wait, t_j = 0 \mid \sum C_0$ e $E\big|p_j = p, no - wait, t_j = 0 \mid \sum T_o$, sendo $\sum C_0$ é o tempo de conclusão e $\sum T_o$ é a soma dos atrasos dos *jobs*.

Chen e Lee [16], assim como o presente estudo, consideraram o chamado problema de sequenciamento de *flow-shop crossdocking* com duas máquinas, cujo objetivo é sequenciar os caminhões de chegada e saída de forma a minimizar o *makespan* (tempo de conclusão do último *job* [33]). A modelagem é feita como um problema de *flowshop* com duas máquinas, porém com restrições de precedência, a fim de ga-

rantir que o caminhão de saída seja processado depois que todas as tarefas de seus predecessores sejam concluídas. Além disso, preempção não é permitida, considera-se que todos os caminhões estão disponíveis no início do horizonte de planejamento e que estoque temporário pode ser formado até a chegada do caminhão de saída apropriado. Os autores provam que esse problema é fortemente NP-difícil e apresentam uma heurística baseada no algoritmo de Johnson e um algoritmo de *branch-and-bound*. Os resultados mostram que o problema pode resolver instâncias com até 60 caminhões em tempo aceitável.

Lima [27], baseado na proposta de Chen e Lee [16], trata do problema de sequenciamento denotado $F2|CD|\sum C_j^2$, ou seja, um problema de *flow-shop* com duas máquinas, com restrições de *crossdocking*, no qual a função objetivo é minimizar a soma das datas de conclusão dos *jobs*. O autor propõe um modelo de programação inteira com formulação baseada em indexação no tempo e em seguida implementa os métodos exatos de relaxação Lagrangeana, Geração de Colunas e dois métodos heurísticos visando obter limites próximos da solução ótima do problema. Os resultados obtidos com os métodos se mostraram satisfatórios. O autor sugere, como futuros trabalhos, a exploração do método de relaxação Lagrangeana, focando na melhora de seu desempenho.

Chen e Song [17] estendem o trabalho de Chen e Lee [16] considerando múltiplas docas de entrada e saída, denominando problema de *crossdocking* híbrido de dois estágios. Nesse caso múltiplos veículos podem ser carregados ou descarregados ao mesmo tempo, considerando que as docas são como máquinas paralelas. Os autores apresentam um modelo de programação inteira mista e propõem heurísticas baseadas no algoritmo de Johnson para solucionar o problema. Em ambos os trabalhos $\beta = t_j = 0$, ou seja, o tempo de transbordo é simplificado e considerado como igual a zero. O modelo de Chen e Lee [16] segue a classificação E2 $\mid t_j = 0 \mid C_{max}$, já Chen e Song [17] modelam o problema como E $\mid t_j = 0 \mid C_{max}$, portanto a diferença está na forma de trabalhar com as docas.

Posteriormente com base nesses dois trabalhos científicos, Araújo e Melo [3] analisam e avaliam heurísticas já existentes na literatura para o problema em questão e propõe outras, objetivando determinar aquelas que obtêm melhores GAP's quando comparadas a um limite inferior. Lira [28] 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. [30] 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 centro de *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 propõem 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 [43] propõem um modelo para o problema de sequenciamento de caminhões 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 e 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.

Li et al. [34] analisaram o problema de sequenciamento de recursos em um centro de *crossdocking* com uma aproximação baseada em *just-in-time*. O problema é modelado como um problema de sequenciamento de máquinas e resolvido com heurísticas, pois é um problema NP-difícil. Os autores propõem duas heurísticas envolvendo algoritmos genéticos que apresentam bons resultados quando comparadas com o desempenho do otimizador CPLEX.

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 tem-

porários 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 centros *crossdocking* em fábricas de manufatura, em que o centro 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 programação dinâmica, onde é feito um estudo detalhado dos limites do espaço de solução.

Alguns trabalhos científicos cujo foco se encontra em problemas relacionados ao sequenciamento de caminhões para CCD são citados por Boysen e Fliedner [12] e encontram-se resumidos na Tabela 2.1.

Esta dissertação de mestrado baseia-se no trabalho científico de Chen e Lee [16] e propõe uma continuidade ao trabalho desenvolvido por Lima [27]. O presente estudo trata o problema exatamente como estudado por Chen e Lee [16] com a finalidade de ampliar e melhorar os resultados.

Tabela 2.1: Histórico de pesquisas relacionadas ao sequenciamento de caminhões em CCD (Fonte: Adaptado de Boysen e Fliedner [12]).

Publicação	Notação do Problema	Complexidade	Contribuição
McWilliams et al. [29]	$[E\|t_{io}\|C_{max}]$	Desconhecida	HM, S
Boysen [11]	$[E\|p_j = p, no-wait, t_j = 0\|\sum T_o]$	Desconhecida	M, HM, E
Boysen et al.	$[E2\|p_j = p, change\|C_{max}]$	NP-difícil	M, HS, HI, E, P
Chen e Lee [16]	$[E2\|t_j = 0\|C_{max}]$	NP-difícil	B, E, P
Chen e Lee [16]	$[E2\|t_j = 0, p_j = p\|C_{max}]$	NP-difícil	P
Chmielewski	$[EM\|r_j, d_j, limit, doors, t_io\|*]$	Desconhecida	M, E
Miao et al. [30]	$[M\|limit, t_{io}\|*]$	NP-difícil	M, HM, P
Boysen [11]	$[M1\|p_j = p, t_j = 0, change\|\sum S_p]$	NP-difícil	M, HM, E, P
Yu e Egbelu [43]	$[E2\|change\|C_max]$	Desconhecida	M, HS
Chen e Song [17]	$[E\|t_j = 0\|C_{max}]$	NP-difícil	M, HS, B
Boysen e Fliedner [12]	$[E\|t_{io}, fix\|\sum w_sU_s]$	NP-difícil	M, P
Boysen e Fliedner [12]	$[E\|t_i = 0, fix\|\sum w_sU_s]$	NP-difícil	P
Araújo e Melo [3]	$[F2(P)\|CD\|C_{max}]$	NP-difícil	HM
Lira [28]	$[F2(P)\|CD\|C_{max}]$	NP-difícil	HM
Lima [27]	$[F2\|CD\|\sum C_j^2]$	NP-difícil	M, HS, HM

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

Capítulo 3

Formulação do problema

Neste capítulo é apresentada uma formulação matemática baseada no problema de sequenciamento proposto iniciamente por Chen e Lee [16], e posteriormente estudada por Lima [27]. O modelo de programação inteira considerado neste trabalho adota a formulação de indexação no tempo proposta por Lima [27] e trata a função objetivo como proposto por Chen e Lee [16], cujo critério de otimização é minimizar o *makespan* (C_{max}).

3.1 Definição do problema

Em um centro de *crossdocking* no qual chegam n caminhões, cada um carregado com produtos que são demandados por um ou vários clientes, deve-se descarregar os produtos desses caminhões e carregá-los em m caminhões de saída, estes são responsáveis por carregar vários tipos de produtos para destinos específicos. Cada caminhão de saída só pode deixar o centro de *crossdocking* se todos os produtos necessários para aquele caminhão já tiverem sido descarregados pelos caminhões de chegada correspondentes em algum momento. Com relação ao número de docas no centro de *crossdocking*, considera-se que existem duas, uma doca de entrada (máquina 1 - M1) e outra de saída (máquina 2 - M2). Dessa forma, o problema é sequenciar o descarregamento dos caminhões de chegada e o carregamento dos caminhões de saída de forma a minimizar o *makespan* (C_{max}), ou seja, minimizar a data de conclusão do último *job* processado pela máquina 2.

Este problema foi proposto inicialmente por Chen e Lee [16], no qual o problema de sequenciamento de caminhões em um centro de *crossdocking* é definido como uma extensão do problema de *flowshop* com duas máquinas em série (*two-machine crossdocking flow shop problem*). Nessa abordagem, a doca de entrada é considerada

como uma máquina (M1) que realiza a operação de descarregamento, os caminhões de chegada são *jobs* que devem ser processados em M1, e a doca de saída é considerada como uma máquina (M2) que realiza a operação de carregamento, os caminhões de saída são *jobs* que devem ser processados por M2. Este problema é considerado como NP-difícil [16].

A particularidade que torna o problema de *crossdocking* diferente do problema de *flowshop* clássico é a introdução de um conjunto de restrições de precedência denominadas restrições de *crossdocking*: um *job* só pode ser processado em M2 se todos os jobs precedentes, representando os caminhões de chegada que contêm os produtos demandados pelo caminhão de saída, tiverem sido processados em M1.

3.2 Modelo Matemático

Como a modelagem do problema será baseada em um problema de sequenciamento, será utilizada uma adaptação da notação adotada por Pinedo [33], que consegue capturar a estrutura de diversos modelos considerados na literatura.

O problema foi modelado como $F2|CD|C_{max}$, ou seja, é um problema de *flowshop* com duas máquinas, com restrições de *crossdocking*, no qual a função objetivo é minimizar a data de conclusão do último *job* processado pela máquina 2.

O modelo estudado neste trabalho adota uma formulação de indexação no tempo proposta por Lima [27], ou seja, defini-se um horizonte de tempo T, esse horizonte é discretizado em $t = 1, 2, ..., T$ períodos. Em cada período t, é tomada a decisão se algum *job* inicia seu processamento em alguma máquina ou não. Esse tipo de formulação gera um número muito elevado de variáveis (número de máquinas x número de *jobs* x T), porém gera limites mais fortes do que outros tipos de formulações encontradas na literatura.

Para a formulação apresentada a seguir os seguintes conjuntos, parâmetros e variáveis são considerados.

Conjuntos

Para representar as chegadas e saídas de caminhões do centro de *crossdocking*, dois conjuntos de *jobs* são criados:

$J^1 = \{j_1^1, j_2^1, ..., j_n^1\}$ representa os caminhões de chegada, que devem ser processados em M1.

$J^2 = \{j_1^2, j_2^2, ..., j_m^2\}$ representa os caminhões de saída, que devem ser processados em M2.

As restrições de *crossdocking* são representadas pela condição abaixo:

Para cada *job* $j_j^2 \in J^2$, existe um subconjunto S_j de *jobs* em J^1, tal que j_j^2 só pode ser processado em M2 se todos os *jobs* em S_j tiverem sido concluídos em M1. Considera-se que cada elemento do subconjunto S_j apresente pelo menos um elemento, ou seja, um determinado *job* $j \in J^2$ possui ao menos um *job* $j \in J^1$ como precedente.

Parâmetros de Entrada

- n: Número de *jobs* a serem processados na máquina 1.
- m: Número de *jobs* a serem processados na máquina 2.
- p_{ij}: Tempo de processamento do *job* j na máquina i.
- T: Horizonte de tempo considerado - Uma primeira estimativa para o horizonte de tempo é a soma dos tempos de processamento de todos os *jobs*.

Variáveis de Decisão

Na formulação indexada no tempo, para cada *job* j e para cada período t, cria-se uma variável que indica se o *job* j inicia seu processamento em t. Para as máquinas M1 e M2, tem-se os dois conjuntos de variáveis respectivamente:

- A variável x_{jt} $(\forall j \in J^1, \forall t \in T)$, será igual a 1, se o *job* j começar no instante t. Caso contrário, o valor da variável será igual a 0.
- A variável y_{jt} $(\forall j \in J^2, \forall t \in T)$, será igual a 1, se o *job* j começar no instante t. Caso contrário, o valor da variável será igual a 0.

Ou seja, o valor da variável x será igual a 1, se o *job* j, $\forall j \in J^1$, inicia seu processamento na máquina M1 no período t. Da mesma forma, a variável y será igual a 1 se o *job* j, $\forall j \in J^2$ inicia seu processamento na máquina M2 no período t. O modelo matemático completo é apresentado a seguir.

$$Min\ \ Z = C_{max} \tag{3.1}$$

sujeito à

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

$$\sum_{t=0}^{T-p_{2j}} y_{jt} = 1, \qquad \forall j \in J^2, \tag{3.3}$$

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

$$\sum_{j \in J^2} \sum_{s=max(0;t-p_{2j}+1)}^{t} y_{js} \leq 1, \qquad \forall t \in T, \tag{3.5}$$

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

$$C_{max} \geq p_{2j} + \sum_{t=0}^{T-p_{2j}} t y_{jt}, \qquad \forall j \in J^2, \tag{3.7}$$

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

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

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

A função objetivo 3.1 minimiza a data de conclusão do último *job* processado pela máquina 2. O conjunto de restrições 3.2 diz que cada *job* $j_j^1 \in J^1$ deve iniciar seu processamento em uma e somente uma data dentro do horizonte de planejamento T. O conjunto 3.3 trabalha com o mesmo raciocínio, porém aplicado aos *jobs* $j_j^2 \in J^2$. As restrições indicadas por 3.4 garantem que um *job* $j_j^1 \in J^1$ não inicia seu processamento enquanto outro *job* estiver sendo processado na máquina 1, da seguinte forma: para cada data $t \in T$, a restrição verifica se algum *job* $j_j^1 \in J^1$ começou a ser processado em uma data maior ou igual a $t - p_{1j} + 1$. Caso afirmativo, j_j^1 ainda está sendo processado, e nenhum *job* pode iniciar seu processamento em t. O conjunto de restrições 3.5 trabalha da mesma forma que o conjunto anterior, no entanto é aplicado aos *jobs* $j_j^2 \in J^2$. O conjunto 3.6 são as restrições de precedência do tipo *crossdocking*. Assim, para cada relação de precedência existente na instância, cria-se uma restrição desse conjunto, na qual a data de início do *job* $j_j^2 \in J^2$ deve ser maior ou igual à data de conclusão de seu precedente $j_k^1 \in S_j$. Se o *job* possuir

mais que um precedente, as restrições relativas aos precedentes, que são processados primeiro, tornam-se redundantes e prevalece aquela relativa ao *job* precedente com maior data de conclusão. O conjunto de restrições 3.7 indicam que a variável C_{max}, *makespan*, deve ser a máxima data de conclusão dos *jobs* processados na máquina 2. Por fim os conjuntos 3.8 à 3.10 definem o domínio das variáveis do modelo.

Capítulo 4

Relaxação Lagrangeana

O modelo da seção 3.2 apresenta um conjunto de restrições definidas como complicantes e que devem ser dualizadas e suas violações penalizadas na função objetivo do novo problema. Na formulação apresentada, o conjunto de restrições a ser relaxado será o conjunto 3.6 que são as restrições de precedência do tipo *crossdocking* e são as restrições de acoplamento entre as variáveis x e y.

O número de restrições desse conjunto é o número de relações de precedência entre um *job* na máquina 2 ($j \in J^2$) e seu precedente na máquina 1 ($k \in S_j$). Por exemplo: Suponha que existem dois *jobs* a serem processados na máquina 2 e cada um deles possui 2 precedentes, o número de relações de precedência é 4. Para cada restrição do conjunto 3.6, associa-se um multiplicador λ, que representará o peso atribuído à violação daquela restrição na função objetivo. Dessa forma, o conjunto de multiplicadores de lagrange do problema será representado por λ_{jk}, sendo ($j \in J^2$) e ($k \in S_j$). Para cada par (j, k), temos a atribuição representada abaixo:

$$\sum_{t=0}^{T-p_{2j}} ty_{jt} - \sum_{t=0}^{T-p_{1k}} (t + p_{1k})x_{kt} \geq 0 \longleftarrow \lambda_{jk} \tag{4.1}$$

Cada restrição do conjunto é representada por $By - Ax \geq 0$. Dessa forma, sua violação é dada por $0 - By + Ax$, ou seja, $Ax - By$, que é a diferença entre a data de conclusão de um precedente na máquina 1 e a data de início de seu sucessor na máquina 2. Considerando a situação em que o *job* na máquina 2 seja sequenciado antes que seu predecessor seja concluído, pois a restrição foi eliminada do problema, essa violação será positiva, e a função objetivo será penalizada. Para cada par (j, k), soma-se na função objetivo o fator $\lambda_{jk}(Ax - By)$, que representa a penalização pela violação daquela restrição em específico. Define-se portanto o subproblema lagrangeano L(λ):

$$L(\lambda) = min\ C_{max} + \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} (\sum_{t=0}^{T-p_{1k}} (t + p_{1k}) x_{kt} - \sum_{t=0}^{T-p_{2j}} t y_{jt}) \tag{4.2}$$

sujeito à

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

$$\sum_{t=0}^{T-p_{2j}} y_{jt} = 1, \quad \forall j \in J^2, \tag{4.4}$$

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

$$\sum_{j \in J^2} \sum_{s=max(0;t-p_{2j}+1)}^{t} y_{js} \leq 1, \quad \forall t \in T, \tag{4.6}$$

$$C_{max} \geq p_{2j} + \sum_{t=0}^{T-p_{2j}} t y_{jt}, \quad \forall j \in J^2, \tag{4.7}$$

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

$$y_{jt} \in 0, 1, \quad \forall j \in J^2 \wedge \forall t \in T, \tag{4.9}$$

$$\lambda_{jk} \geq 0, \quad \forall j \in J^2 \wedge k \in S_j, \tag{4.10}$$

$$C_{max} \geq 0. \tag{4.11}$$

Os passos para a resolução de L(λ) são descrito na seção seguinte.

4.1 Resolvendo o Subproblema Lagrangeano

Analisando o modelo da seção 4, o leitor perceberá que a aplicação da relaxação lagrangeana resultou em um subproblema com algumas características especiais, que tornam possível a decomposição desse subproblema em dois problemas independentes. É possível identificar que com a eliminação do conjunto de restrições 3.6, as variáveis x e y tornam-se desacopladas, dessa forma, não há restrições no novo problema que relacionam os valores de x com os valores de y. Portanto, separando também na função objetivo os termos que contém valores da variável x dos termos que contém valores da variável y, pode-se definir dois subproblemas menores independentes, cada qual com termos da função objetivo e restrições de uma das variáveis. Assim, a solução de cada um dos modelos separadamente surge como uma alternativa. O valor de L(λ), que será um limite inferior para o problema, é obtido

agregando os subproblemas $L(\lambda)_x$ e $L(\lambda)_y$.

Subproblema em X

Isolando os termos da função objetivo que contêm x, chega-se ao subproblema em X:

$$L(\lambda)_x = min \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=0}^{T-p_{1k}} (t + p_{1k}) x_{kt} \tag{4.12}$$

sujeito à

$$\sum_{t=0}^{T-p_{1j}} x_{jt} = 1, \qquad \forall j \in J^1, \tag{4.13}$$

$$\sum_{j \in J^1} \sum_{s=max(0;t-p_{1j}+1)}^{t} x_{js} \leq 1, \qquad \forall t \in T, \tag{4.14}$$

$$x_{jt} \in 0, 1, \qquad \forall j \in J^1 \wedge \forall t \in T. \tag{4.15}$$

Alternativamente, podemos reescrever a função objetivo acima sob a perspectiva dos *jobs* em J^1, ao invés de acessá-los indiretamente através do conjunto de precedentes dos *jobs* em J^2, conforme abaixo:

$$min \sum_{j \in J^1} \sum_{t=0}^{T-p_{1j}} (t + p_{1j}) x_{jt} w_j^1 \tag{4.16}$$

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

Como indicado acima, w_j^1 é a soma das penalidades de todos os *jobs* na máquina 2 que possuem o *job* j como precedente na máquina 1. Esse valor é uma constante que representa o peso do *job* j, e será maior se este *job* for responsável por grandes violações das restrições de precedência.

O problema citado acima é conhecido como Tempo Total de Conclusão Ponderado ([33]), denotado por $1||\sum C_j W_j$, sendo $C_j = (t + p_{1j}) x_{jt}$ e $W_j = w_j^1$. Dessa forma o subproblema em X pode ser reescrito como:

$$min \sum_{j \in J^1} C_j W_j \tag{4.17}$$

Segundo Pinedo([33]) esse problema pode ser resolvido por ordenação de acordo com

a regra WSPT (*Weighted Shortest Processing Time First*) proporta por Smith em 1956 [37]. De acordo com essa regra, a solução ótima é obtida ordenando-se os *jobs* em ordem decrescente de w_j^1 / p_{1j}, e pode ser obtida em O($nlog(n)$).

Proposição 4.1.1 *A regra WSPT é ótima para o problema* $1||\sum C_j W_j$.

Demonstração Suponha, por contradição, uma sequência ótima S, não seguindo a regra WSPT, na qual existem dois *jobs* adjacentes j e k, sendo j seguido de k, tal que $w_j^1/p_{1j} < w_k^1/p_{1k}$, $w_j^1 > 0$ $\forall j$. Dessa forma, o tempo total de conclusão ponderado T_{jk} dos *jobs* j e k é (t + p_{1j})w_j^1 + (t + p_{1j} + p_{1k})w_k^1. Troca-se então j e k de posição, mantendo os demais *jobs* em suas posições originais, denominando a nova sequência S', onde o tempo total de conclusão ponderado T_{kj} dos *jobs* j e k é (t + p_{1k})w_k^1 + (t + p_{1k} + p_{1j})w_j^1. Como w_j^1 / $p_{1j} < w_k^1$ / p_{1k}, verifica-se que sob S', T_{kj} é sempre menor que T_{jk} em S, o que contradiz a otimalidade de S, provando que a regra WSPT chega a solução ótima do problema. ∎

Subproblema em Y

Com os termos que contém y isolados da função objetivo do subproblema lagrangeano, obtém-se o subproblema em Y:

$$L(\lambda)_y = min\ C_{max} - \sum_{j\in J^2}\sum_{k\in S_j}\lambda_{jk}\sum_{t=0}^{T-p_{2j}} t y_{jt} \tag{4.18}$$

sujeito à

$$\sum_{t=0}^{T-p_{2j}} y_{jt} = 1, \quad \forall j \in J^2, \tag{4.19}$$

$$\sum_{j\in J^2}\sum_{s=max(0;t-p_{2j}+1)}^{t} y_{js} \leq 1, \quad \forall t \in T, \tag{4.20}$$

$$C_{max} \geq p_{2j} + \sum_{t=0}^{T-p_{2j}} t y_{jt}, \quad \forall j \in J^2, \tag{4.21}$$

$$y_{jt} \in 0,1, \quad \forall j \in J^2 \wedge \forall t \in T, \tag{4.22}$$

$$\lambda_{jk} \geq 0, \quad \forall j \in J^2 \wedge k \in S_j, \tag{4.23}$$

$$C_{max} \geq 0. \tag{4.24}$$

Definindo $w_j^2 = -\sum_{k\in S_j}\lambda_{jk}$ $\forall j \in J^2$, podemos reescrever o problema da seguinte maneira:

$$L(\lambda)_y = min\ C_{max} + \sum_{j \in J^2} \sum_{t=0}^{T-p_{2j}} t.y_{jt} w_j^2 \tag{4.25}$$

s.t. (4.19), (4.20), (4.21), (4.22) e (4.24).

O subproblema em Y acima pode ser considerado um problema de sequenciamento em máquina única. O primeiro termo da função objetivo acima representa a maior data de conclusão na máquina 2. O segundo termo representa o somatório das datas de início ponderadas dos *jobs* na máquina 2, este problema é conhecido como Tempo Total de Início Ponderado, denotado $\sum I_j W_j$, sendo $I_j = t.y_{jt}$ e $W_j = w_j^2$ referente ao peso do *job* j na máquina 2. Dessa forma, o subproblema em Y pode ser definido como $1||C_{max} + \sum I_j W_j$. Para resolver este subproblema, criamos uma regra denominada WSPT-TRD capaz de alocar os *jobs* de maneira correta, conforme pseudocódigo abaixo.

A regra WSPT-TRD funciona da seguinte maneira, primeiramente aplica-se a regra proposta por Smith em 1956 [37] para gerar a sequência de *jobs* na máquina 2. Obtida a sequência, o próximo passo consiste em definir a maneira correta de alocar os *jobs* no horizonte de tempo. Para isso foi necessário realizar uma avaliação da expansão do *makespan* C_{max} quando os pesos associados aos *jobs* são negativos. Essa avaliação é de extrema importância para a eficácia da regra, uma vez que, quando os pesos são negativos, cria-se uma situação de *trade-off* na função objetivo, conforme explicado a seguir.

A função objetivo do subproblema em Y estudado é composta por dois termos, o *makespan* C_{max} e $\sum I_j W_j$. Analisando a situação em que os pesos dos *jobs* são positivos (no nosso caso $w_j^2 = 0$, dado que $\lambda_{jk} \geq 0$), tanto C_{max} quanto $\sum I_j W_j$ possuem comportamento similar, ou seja, dado que a função objetivo é de minimização, a solução ótima é encontrada alocando-se os *jobs* no início do horizonte de tempo de forma a obter o menor valor possível para a função objetivo.

O *trade-off* aparece quando os pesos dos *jobs* são negativos ($w_j^2 < 0$). Nesse caso os termos da função objetivo possuem comportamento divergente, de um lado o *makespan* C_{max} tende a alocar os *jobs* no início do horizonte de tempo, e do outro o $\sum I_j W_j$ tende a alocar os *jobs* o mais tarde possível, uma vez que, para pesos negativos, isso faria com que a função objetivo decrescesse. Para lidar com esse *trade-off* analisou-se a taxa de crescimento de C_{max} quando deslocamos os *jobs* com peso negativo mais para o final do horizonte. Verificamos que a taxa de crescimento/expansão de C_{max} é constante e igual a 1, ou seja, independentemente do tamanho do deslocamento dos *jobs* no horizonte de tempo, C_{max} sempre seria acrescido de uma unidade. Assim criamos uma variável denominada Negsoma para o

cálculo da soma dos pesos negativos, pois, se a soma dos pesos negativos é menor que -1, o deslocamento dos *jobs* mais para o fim do horizonte compensa o aumento de C_{max}, definindo a melhor maneira de alocar os *jobs*. Essa situação de *trade-off* pode ser mais bem compreendida através do exemplo exposto na Figura 4.1 a seguir.

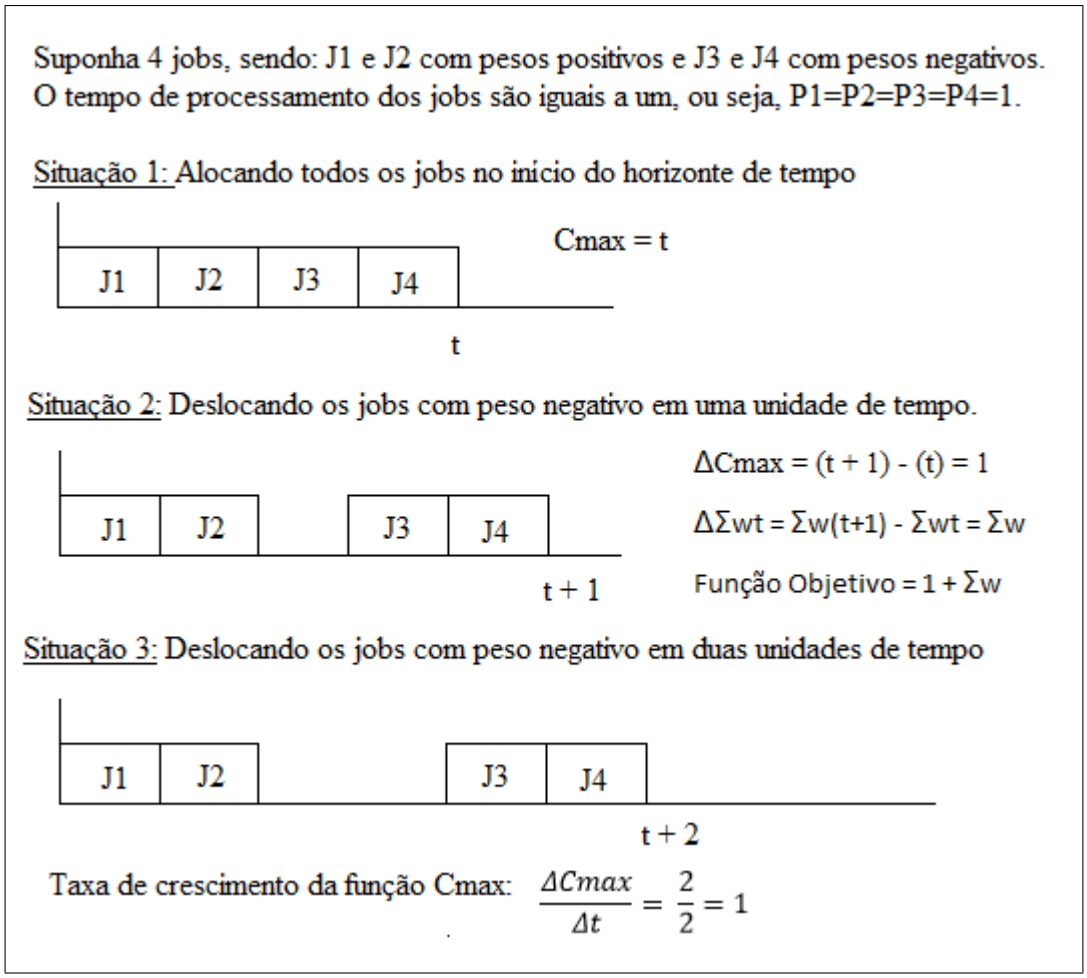

Figura 4.1: Exemplo da expansão do C_{max} com o deslocamento dos *jobs*

Pseudocódigo da regra WSPT-TRD:

1. Calcule os pesos na máquina 2, sendo $w_j^2 = -\sum_{k \in S_j} \lambda_{jk}$.
2. Obtenha a sequência ordenando-se os *jobs* em ordem decrescente de w_j^2 / p_{2j}, conforme a regra WSPT de Smith.
3. Se existir pesos negativos ($w_j^2 < 0$), calcule a soma dos pesos negativos através de Negsoma $= \sum_{j \in J^2} w_j^2$.

   O cálculo da soma dos pesos negativos é importante, pois define a forma adequada de alocação dos *jobs*, de forma a compensar o *trade-off* entre *makespan* e tempo total de início ponderado quando $w_j^2 < 0$.
4. Se $w_j^2 = 0$ ou Negsoma ≥ -1, aloque os *jobs* a partir de $t = 0$, de frente para

trás, conforme sequência gerada pelo WSPT. No caso estudado como $\lambda_{jk} \geq 0$, então não temos $w_j^2 > 0$, mas o método criado aplica-se também para $w_j^2 > 0$.

5. Se $w_j^2 < 0$, aloque os *jobs* a partir de t = T, de trás para frente, mantendo a sequência gerada pela regra WSPT.

6. Calcule C_{max} como a maior data de término.

7. Calcule a função objetivo igual a $C_{max} - \sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk} \sum_{t=0}^{T-p_{2j}} t.y_{jt}$.

Para provar que o problema $1||C_{max} + \sum I_j W_j$ pode ser resolvido por ordenação segundo a regra WSPT-TRD, temos que demonstrar duas situações:

1. Demonstrar que $1||\sum I_j W_j$ é equivalente a $1||\sum C_j W_j$.

2. Demonstrar que a regra WSPT-TRD é ótima para o problema $1||C_{max} + \sum I_j W_j$.

Proposição 4.1.2 *O problema* $1||\sum I_j W_j$ *é equivalente a* $1||\sum C_j W_j$.

Demonstração O problema $1||\sum I_j W_j$ é equivalente a $1||\sum C_j W_j$, uma vez que:

$$\sum I_j W_j = \sum (I_j + p_j - p_j) W_j = \sum (C_j - p_j) W_j = \sum C_j W_j - \sum p_j W_j$$

Analisando o último termo vemos que $\sum p_j W_j$ é uma constante, então podemos concluir que o problema $1||\sum I_j W_j$ é equivalente a $1||\sum C_j W_j$, e pode ser resolvido pela regra WSPT. ∎

Proposição 4.1.3 *A regra WSPT-TRD é ótima para o problema* $1||C_{max} + \sum I_j W_j$.

Demonstração A demonstração da regra será dividida em duas partes.

Parte I: Quando $w_j^2 \geq 0$.

Suponha, por contradição, uma sequência ótima S, não seguindo a regra WSPT, na qual existem dois *jobs* adjacentes j e k, sendo j seguido de k, tal que $w_j^2/p_{2j} < w_k^2/p_{2k}$, $w_j^2 \geq 0$ $\forall j$. Dessa forma, o tempo total de conclusão ponderado T_{jk} dos *jobs* j e k é $(t + p_{2j})w_j^2 + (t + p_{2j} + p_{2k})w_k^2$. Troca-se então j e k de posição, mantendo os demais *jobs* em suas posições originais, denominando a nova sequência S', onde o tempo total de conclusão ponderado T_{kj} dos *jobs* j e k é $(t + p_{2k})w_k^2 + (t + p_{2k} + p_{2j})w_j^2$. Ao realizar a troca de j e k o *makespan* C_{max} não se altera. Como $w_j^2 / p_{2j} < w_k^2 / p_{2k}$, verifica-se que sob S', T_{kj} é sempre menor que T_{jk} em S, o que contradiz a otimalidade de S, provando que a regra WSPT chega a solução ótima do problema.

Neste caso, se os $w_j^2 \geq 0$ deve-se sequenciar os *jobs* a partir de $t = 0$.

Parte II: Quando $w_j^2 < 0$.

Quando $\sum_{k \in S_j} \lambda_{jk} < 0$ para determinado j, w_j^2 é negativo, contrariando a suposição que $w_j^2 \geq 0 \ \forall \ j$. Neste caso, deve-se alocar os *jobs* com $w_j^2 < 0$ a partir de t = T, mantendo a sequência gerada pela regra WSPT.

Como a função objetivo é de minimização, deve-se associar um grande valor para t, ou seja, a data de início, para *jobs* com peso negativo, e um valor baixo de t para *jobs* com peso positivo ou com a soma dos pesos negativos maior ou igual a -1, minimizando assim a função objetivo composta pelo *makespan* e pela soma das datas de início ponderadas. ∎

4.2 Resolvendo o Dual Lagrangeano

Definido o subproblema lagrangeano é preciso resolver o dual lagrangeano, ou seja, encontrar o conjunto de pesos λ que maximize o limite inferior para o problema inteiro. Existem dois métodos muito utilizados na literatura, que são o Algoritmo do Subgradiente e o Algoritmo do Volume. Neste trabalho, o dual lagrangeano foi resolvido pelo Algoritmo do Volume conforme apresentaremos na próxima seção. Optamos pela utilização do Algoritmo do Volume pois segundo Vanderbeck e Wolsey [41] o desempenho de convergência do algoritmo do volume é tipicamente melhor do que o do algoritmo do subgradiente.

4.2.1 Algoritmo do Volume

O dual Lagrangeano L(λ) foi resolvido pelo Algoritmo do Volume tal como proposto no trabalho do Nogueira em [31] e pelo autores Barahona e Anbil em [5]. O desempenho do algoritmo é analisado por Barahona e Ladányi em [6] e por Fukuda em [19]. O Algoritmo do Volume é uma extensão do método do subgradiente, que produz uma aproximação de solução primal enquanto se resolve o dual, assim, é capaz de provar a otimalidade. Este algoritmo tem desempenho computacional semelhante ao algoritmo do subgradiente e apresenta semelhanças com o Método do Subgradiente Conjugado [25],[42] e Método de Feixe [26], [24]. A convergência global de uma implementação específica do algoritmo do volume é analisada por Bahiense et al. em [4].

No algoritmo proposto as restrições de precedência são dualizadas. Os multiplicadores de Lagrange definem um peso para os *jobs* alocados em uma dada posição. Se o peso do *job* é um valor grande, significa que ele tem um maior impacto na função

objetivo, pois é grande a violação sobre a restrição dualizada, ou seja, o *job* está alocado em uma posição desvantajosa. Essa informação é importante e pode ser usada para decidir quando sequenciar os *jobs*.

Para aplicação do algoritmo considere:

x: solução do Subproblema em X

y: solução do Subproblema em Y

z: valor da função objetivo do subproblema lagrangeano

λ: multiplicadores de Lagrange obtidos pelo Algoritmo do Volume

$\nu(\lambda, x, y)$: subgradiente

UB: limite superior ou solução viável.

O UB é encontrado através das heurísticas construtivas polinomiais tal como proposto na seção 4.3.

Os passos do algoritmo proposto podem ser resumidos conforme a seguir:

Passo 0: Inicialize um vetor λ_{jk}, onde λ_{jk}=0. Resolva o subproblema lagrangeano e obtenha a solução inicial do Subproblema em X (x^0), do Subproblema em Y (y^0) e o valor da função objetivo (z^0). Obtenha um UB através da heurística polinomial e compute $\bar{UB}$ sendo a melhor solução viável obtida pelas heurísticas polinomiais. Faça $\bar{x}$=x^0, $\bar{y}$=y^0, $\bar{z}$=z^0 e k=1.

Passo 1: Compute o subgradiente $\nu(\lambda_{jk-1}, \bar{x}, \bar{y})$ e λ_{jk}=$\bar{\lambda_{jk}}$ + $s\nu(\lambda_{jk-1})$ o cálculo do tamanho do passo s é dado pela equação 4.29. Resolva o subproblema lagrangeano com o novo λ_{jk} e obtenha as soluções x^k, y^k e z^k. Então faça $\bar{x}$=αx^k + (1-α)$\bar{x}$, $\bar{y}$=αy^k + (1-α)$\bar{y}$ e $\bar{\lambda}$=$\alpha\lambda^k$ + (1-α)$\bar{\lambda}$, onde α é um número entre 0 e 1, definido por combinação convexa tal como definido em 4.26.

Passo 2: Se $z^k > \bar{z}$ atualize $\bar{\lambda}$=λ_{jk} e $\bar{z}$=z^k. Se $\bar{z}$ é melhorado em 10% vá para o Passo 3. Senão vá para o Passo 4.

A proposta de executar as heurísticas construtivas cada vez que o limite inferior melhora em 10% baseou-se no trabalho de Paula et al. em [32], segundo os autores testes preliminares mostraram que os valores e as soluções do problema relaxado mudam relativamente pouco de uma iteração para outra, a execução das heurísticas com mais frequência seria dispendioso e levaria à mesma solução com muita freqüência. Ainda segundo os autores uma busca local após cada execução das heurísticas, em vez de executar as heurísticas mais freqüentemente, levaria a melhores soluções finais.

Passo 3: Heurísticas Construtivas Polinomiais para obtenção de um Limite Superior. Nesse passo as quatro heurísticas são testadas e o melhor valor encontrado para o limite superior computado. Além disso, se o limite superior for melhorado, adicionamos cortes no horizonte, reduzindo o horizonte de tempo, que passa a receber o valor do melhor limite superior encontrado até o momento. Dessa forma conseguimos melhorar os limites, tanto superior quanto inferior, e aumentar o desempenho do algoritmo a cada iteração.

Passo 4: Critério de Parada: se satisfeito então pare. Senão k = k + 1 e volte ao Passo 1.

O tamanho do passo s e o parâmetro α usados para definir $\bar{x}$ e $\bar{y}$ são calculados tal como proposto em [5] e [19]. Primeiro definimos α_{opt} da seguinte forma:

$$\alpha_{opt} = argmin \parallel \alpha\nu'^{k}_{(\lambda_{jk-1},x^k,y^k)} + (1-\alpha)\nu^{k}_{(\lambda_{jk-1},\bar{x},\bar{y})} \parallel^2 \tag{4.26}$$

Os valores dos parâmetros π, MaxWaste, factor, α_{max}, st, α, amarela e verde, foram definimos através da utilização do método SPOT, conforme explicado no Apêndice A. Os parâmetros α_{max} e α são inicialmente definidos como 0.3033703759 e 0.0830043818, respectivamente, baseado nos testes computacionais realizados através do SPOT, e α é computado como:

$$\alpha = \alpha_{max} * \alpha \quad se \quad \alpha_{opt} < 0 \tag{4.27}$$

$$\alpha = min\{\alpha_{opt}, \alpha_{max}\} \quad se \quad \alpha_{opt} \geq 0 \tag{4.28}$$

Antes de definirmos o tamanho do passo s, precisamos definir o parâmetro π. Portanto, para definir o valor de π definimos três tipos de iterações baseado nos trabalhos de Barahona e Anbil [5] e Barahona e Ladányi [6].

Cada iteração sem melhoria é denominada iteração vermelha, dessa forma o número máximo de iterações sem melhoria do limite inferior é definido pelo parâmetro MaxWaste. Se $z^k > \bar{z}$ e $\nu'^{k}_{(\lambda_{jk-1},x^k,y^k)}\nu^{k}_{(\lambda_{jk-1},\bar{x},\bar{y})} < 0$ tal como mencionado por Barahona e Anbil [5], significa que um passo maior em direção a ν^k resultaria em um valor menor para z^k. Essas iterações são denominadas amarela, caso contrário, a iteração é denominada verde. A cada iteração amarela multiplicaremos π pelo parâmetro amarela. A cada iteração verde multiplicaremos π pelo parâmetro verde. Depois de 24 (MaxWaste) iterações vermelhas consecutivas, multiplique π pelo parâmetro factor. Assim, o tamanho do passo s na iteração k, é definida como:

$$s = \frac{\pi * (st * UB - \bar{z}))}{\parallel \nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})} \parallel^2} \tag{4.29}$$

Finalmente o critério de parada do Algoritmo do Volume é determinado se um dos seguintes critérios é satisfeito:

1. Número máximo de iterações, neste caso 1000 iterações, ou;
2. Número de iterações sem melhoria de limite inferior, definido por: $\frac{z^k > \bar{z}}{\bar{z}} < 0.1$, ou;
3. Módulo do subgradiente nulo ou desprezível: $\parallel \nu_{(\lambda_{jk-1}, \bar{x}, \bar{y})} \parallel^2 \leq 0.000001$.

4.3 Heurísticas Construtivas Polinomiais para obtenção de um Limite Superior

Simultaneamente a obtenção de um limite inferior deve-se obter uma solução viável no espaço de soluções do problema original, uma vez que a eliminação das restrições de precedência torna o espaço de soluções maior, e uma solução encontrada nesse novo espaço não é necessariamente viável no espaço anterior. Encontrada essa solução, seu valor da função objetivo será um limite superior para o problema original.

A principal questão na obtenção de um limite superior é encontrar uma solução próxima da solução ótima do problema, representando um limite superior de maior qualidade dentro do procedimento de resolução, o que proporcionaria maior rapidez na obtenção da solução ótima.

Nessa seção são detalhadas as heurísticas construtivas polinomiais implementadas para obtenção de um limite superior do problema $F2|CD|C_{max}$ para instâncias de pequeno e grande porte. Primeiramente será apresentada a heurística 1, que considera a sequência na máquina 1 fixa, obtida através da resolução do subproblema X, e gera a sequência na máquina 2 a partir do cálculo da *release date* dos *jobs*. Posteriormente será apresentada a heurística 2, que considera a sequência na máquina 2 fixa, obtida através da resolução do subproblema Y e gera a sequência na máquina 1 respeitando as relações de precedência conforme a sequência na máquina 2. As heurísticas 3 e 4 são versões integradas das heurísticas 1 e 2. A heurística 3 usa as informações dos multiplicadores de lagrange para ordenar os *jobs* na máquina 1 e a heurística 4 usa as informações dos multiplicadores de lagrange para ordenar os *jobs* na máquina 2. As quatro heurísticas propostas serão apresentadas a seguir. De

forma que nos capítulos seguintes essas heurísticas sejam comparadas.

Heurística 1 Proposta

1. Obtenha a sequência na máquina 1 através da resolução do subproblema em X, ou seja, ordenando-se os *jobs* dessa máquina de acordo com a regra WSPT.
2. Calcule, para cada *job* da máquina 2, sua *release date* (r_j, $j \in J^2$).

   A *release date* de um *job* j representa a data a partir da qual aquele *job* pode ser sequenciado. É calculada como $r_j = max_{k \in S_j} C_k$, ou seja, a maior data de conclusão dentre os precedentes de j na máquina 1 na sequência dada. A sequência obtida deve minimizar o *makespan*, que é a função objetivo do problema original.
3. Obtenha a sequência na máquina 2 ordenando-se os *jobs* em ordem não decrescente de *release dates* (r_j).

Heurística 2 Proposta

1. Obtenha a sequência na máquina 2 através da resolução do subproblema em Y.
2. Sequencie os *jobs* na máquina 1 conforme a sequência obtida pelo subproblema em Y, respeitando as relações de precedência de *crossdocking*.

   Caso exista um *job* na máquina 1 sem relação de precedência na máquina 2, sequencie esse *job* por último.
3. Calcule as novas *release dates* dos *jobs* na máquina 2, a partir da sequência na máquina 1.
4. Obtenha a sequência na máquina 2 ordenando-se os *jobs* em ordem não decrescente de *release dates* (r_j).

Heurística 3 Proposta

1. Para cada *job* na máquina 1 ($k \in J^1$), calcule um $\beta_k = \dfrac{\sum_{j \in J^2} \sum_{k \in J^1} \lambda_{jk}}{Nprec_k}$.

   Onde $Nprec_k$ corresponde ao número de vezes que *job* k da máquina 1 é precedente na máquina 2.
2. Obtenha a sequência na máquina 1 ordenando-se os *jobs* em ordem decrescente de β_k.

3. Calcule as novas *release dates* dos *jobs* na máquina 2, a partir da sequência na máquina 1.
4. Obtenha a sequência na máquina 2 ordenando-se os *jobs* em ordem não decrescente de *release dates* (r_j).

Heurística 4 Proposta

1. Para cada *job* na máquina 2 ($j \in J^2$), calcule um $\beta_j = \dfrac{\sum_{j \in J^2} \sum_{k \in S_j} \lambda_{jk}}{Nprec_j}$.

   Onde $Nprec_j$ corresponde ao número de precedentes que o *job* j possui.
2. Obtenha a sequência na máquina 2 ordenando-se os *jobs* em ordem crescente de β_j.
3. Sequencie os *jobs* na máquina 1 conforme a sequência na máquina 2, respeitando as relações de precedência de *crossdocking*.

   Caso exista um *job* na máquina 1 sem relação de precedência na máquina 2, sequencie esse *job* por último.
4. Calcule as novas *release dates* dos *jobs* na máquina 2, a partir da sequência na máquina 1.
5. Obtenha a sequência na máquina 2 ordenando-se os *jobs* em ordem não decrescente de *release dates* (r_j).

Capítulo 5

Experimentos Computacionais

Os experimentos realizados neste trabalho foram executados utilizando-se uma máquina com processador Intel (R) Xeon (R) CPU X5690 @ 3.47GHz com 24 processadores, 132 GB de RAM, e sistema operacional Ubuntu Linux. Foi utilizada a linguagem de programação C++ e o software de otimização CPLEX 12.4.

5.1 Geração de Instâncias

As instâncias utilizadas nesse trabalho foram geradas através do software MATLAB, baseando-se nas propriedades das instâncias utilizadas por Chen e Lee [16]. As propriedades das instâncias geradas e aplicadas no modelo foram:

- Existem dois grupos de instâncias: o primeiro possui *jobs* curtos, cujos tempos de processamento foram gerados uniformemente entre 1 e 10. O segundo grupo apresenta *jobs* longos com tempos de processamento gerados uniformemente entre 10 e 100. Essa variação nos tempos de processamento tem como objetivo avaliar a influência do tamanho do horizonte de tempo no desempenho do método de resolução.
- Para cada grupo de instância o número de *jobs* na máquina 1 é fixo e igual a 5, 10, 20, 40 ou 60. O número de *jobs* na máquina 2 varia, sendo $n\alpha$, onde α = [0.6, 0.8, 1.0, 1.2, 1.4].
- Para cada caso, 10 diferentes problemas são resolvidos. O resultado apresentado na Tabela 5.2 representa a média dos 10 casos.
- O número máximo de precedentes de cada *job* na máquina 2 é gerado por meio de uma distribuição uniforme entre 1 e n-1.

Tabela 5.1: Sumário das instâncias geradas para teste, separadas em grupo 1 e 2, e subgrupos, de acordo com o número de *jobs* na máquina 1 ($JobsM1(n)$). Além disso, a tabela informa o número de *jobs* na máquina 2 ($JobsM2(m)$), o número máximo de precedentes do *job* $j \in J^2$ (NP), o tempo de processamento dos *jobs* (TP), o código da instância ($Código$) e por último quantas instâncias foram geradas para cada subgrupo ($N.Inst$).

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

Cada subgrupo de instâncias geradas possuirá um código único para facilitar a referência ao longo da apresentação dos resultados e análises, no formato *n-m-np-g*, onde n é o número de *jobs* na máquina 1, m é o número de *jobs* na máquina 2, np é o número máximo de precedentes de cada *job* na máquina 2 e g é o grupo o qual pertence o subgrupo.

A Tabela 5.1 apresenta um resumo das instâncias geradas e suas respectivas características.

Os valores dos tempos de processamento dos *jobs*, presentes nos grupos J^1 ou J^2, foram definidos de forma aleatória, assim como o número de precendetes, de acordo com a respectiva distribuição uniforme definida na tabela. Os valores que representam n e m foram estabelecidos anteriormente. A configuração definida busca avaliar os resultados quando a quantidade de *jobs* é alterada. Além disso, é analisado as interferências relativas ao tempo de processamento dos *jobs* e ao número de precedentes de um *job* que será processado pela máquina 2.

5.2 Resultados Computacionais

Os resultados dos testes computacionais realizados estão expostos na Tabela 5.2. A coluna Código da Instância refere-se a instância trabalhada, os itens MC, RL e RLg significam modelo completo, relaxação linear e relaxação Lagrangeana, respectivamente. Já as siglas FO, Melhor_LI, LI, LS, GAP e T representam os resultados

médios para a função objetivo, melhor limite inferior encontrado, limite inferior, limite superior, o GAP de otimalidade calulado por $\frac{LS - LI}{LS}$ e o tempo de processamento em segundos, respectivamente.

Para cada instância, 10 diferentes problemas foram resolvidos e a média dos resultados é apresentada na Tabela 5.2. O tempo de execução foi limitado em 1 hora. Se em menos de 1 hora é encontrada a solução ótima do problema, a execução é automaticamente interrompida, caso a solução não seja encontrada em menos de uma hora, a execução é interrompida e os valores arquivados. O traço (-) significa que o valor correspondente não foi encontrado dentro do limite de tempo.

A Tabela 5.3 apresenta o número de vezes que as heurísticas obtiverem o melhor limite superior da iteração. Esses resultados foram gerados a fim de avaliar e comparar o desempenho das heurísticas construtivas polinomiais propostas.

5.3 Análise dos Resultados

Por meio dos resultados expostos na Tabela 5.2, é possível comparar o desempenho do modelo completo, da relaxação linear e da relaxação Lagrangeana, para todas as instâncias testadas.

O experimento mostrou que o modelo proposto é eficiente para resolver instâncias com n=5 e n=10 *jobs* com tempos de processamento curtos, e em sua maioria, em baixo tempo computacional. Não foi identificada a solução ótima para as maiores instâncias do grupo 1 e no grupo 2 apenas duas instâncias tiveram sua solução ótima identificada. Esse fato expõe a dificuldade de resolução de modelos com indexação no tempo. Com a indexação no tempo, o número de variáveis é proporcional ao horizonte de tempo, e quanto maior o número de *jobs* da instância, maior o horizonte de tempo necessário para sequenciá-los. Estes fatores aumentam a complexidade do problema, sendo necessário maior esforço computacional para resolvê-lo.

Ao se comparar as relaxações, verifica-se que a relaxação Lagrangeana fornece melhores limites inferiores do que os encontrados pela relaxação linear, além de ser capaz de resolver todas as instâncias em baixo tempo computacional. Isso foi possível graças a redução que realizamos no horizonte de tempo, sempre que o limite superior é melhorado. Quanto ao limite superior nota-se que os valores encontrados não são muito distantes do valor fornecido pelo modelo completo, principalmente nas instâncias em que a solução ótima foi identificada. Dessa forma, trabalhar na melhora do cálculo dos limites, associando-os as heurísticas, pode ser uma boa opção na busca de soluções ótimas para instâncias maiores.

A Tabela 5.3 apresenta o desempenho das heurísticas polinomiais desenvolvidas para obtenção de um limite superior. O valor apresentado na Tabela representa o número de vezes que as heurísticas obtiverem o melhor limite superior da iteração. Analisando os resultados o leitor pode verificar que a heurística 1 assim como a heurística 2 apresentaram melhores desempenhos, uma vez que encontraram os melhores limites superiores na maioria das iterações. A heurística 4, que utiliza as informações dos multiplicadores de lagrange para ordenar os *jobs* na máquina 2, também apresentou bom desempenho sendo capaz de encontrar uma solução próxima da solução ótima do problema. Já a heurística 3 apresentou desempenho inferior se comparado com as demais heurísticas.

Tabela 5.2: Comparação entre Modelo Completo, Relaxação Linear e Relaxação Lagrangeana. A coluna Código da Instância refere-se a instância trabalhada, os itens MC, RL e RLg significam modelo completo, relaxação linear e relaxação Lagrangeana, respectivamente. Já as siglas FO, Melhor_LI, LI, LS, GAP e T representam a função objetivo, melhor limite inferior encontrado, limite inferior, limite superior, o GAP de otimalidade ($\frac{LS-LI}{LS}$) e o tempo de processamento em segundos, respectivamente. O traço (-) significa que o valor correspondente não foi encontrado.

Código da	MC				RL		RLg			
Instância	FO	Melhor_LI	GAP	T(seg)	FO	T(seg)	LI	LS	GAP	T(seg)
5-3-4-1	33	33	0%	0.2	23	0.0	26	35	22%	0.6
5-4-4-1	33	33	0%	0.3	23	0.0	25	33	22%	0.7
5-5-4-1	34	34	0%	0.5	23	0.0	28	35	19%	0.6
5-6-4-1	39	39	0%	1.3	24	0.0	31	41	24%	0.4
5-7-4-1	44	44	0%	3.7	25	0.0	36	48	23%	0.4
10-6-9-1	59	59	0%	20.2	36	0.0	40	67	39%	0.9
10-8-9-1	66	66	0%	167.5	39	0.1	44	73	39%	0.7
10-10-9-1	71	71	0%	369.6	39	0.1	52	81	35%	0.5
10-12-9-1	82	77	5%	2440.5	40	0.1	70	91	23%	0.2
10-14-9-1	90	86	5%	3141.6	44	0.1	80	104	23%	0.0
20-12-19-1	135	114	15%	3600	65	0.3	67	148	55%	1.1
20-16-19-1	153	102	32%	3600	66	0.7	90	165	46%	1.0
20-20-19-1	167	107	36%	3600	68	0.6	115	182	37%	0.8
20-24-19-1	190	115	38%	3600	72	0.9	139	203	31%	1.1
20-28-19-1	206	130	36%	3600	82	1.0	153	222	31%	0.7
40-24-39-1	306	164	46%	3600	122	2.0	140	309	55%	1.1
40-32-39-1	-	-	-	3600	126	3.5	179	350	48%	1.0
40-40-39-1	-	-	-	3600	124	4.9	220	377	42%	1.0
40-48-39-1	-	-	-	3600	135	12.7	262	413	37%	1.0
40-56-39-1	-	-	-	3600	156	26.2	306	447	31%	1.7
60-36-59-1	-	-	-	3600	176	34.5	198	464	57%	3.8
60-48-59-1	-	-	-	3600	176	43.0	269	521	48%	4.6
60-60-59-1	-	-	-	3600	181	133.1	337	578	42%	3.0
60-72-59-1	-	-	-	3600	203	112.7	396	632	37%	4.9
60-84-59-1	-	-	-	3600	241	122.8	463	696	33%	8.6
5-3-4-2	317	317	0%	21.0	225	0.2	167	338	50%	0.0
5-4-4-2	330	330	0%	354.9	229	0.3	194	352	43%	0.0
5-5-4-2	343	339	1%	1570.1	227	0.4	237	369	34%	0.3
5-6-4-2	393	340	12%	2388.3	237	0.6	287	425	32%	0.1
5-7-4-2	441	376	13%	2882.5	246	1.9	356	485	25%	0.1
10-6-9-2	625	438	29%	3600	355	10.4	366	688	46%	0.4
10-8-9-2	731	454	37%	3600	363	12.4	470	730	35%	0.6
10-10-9-2	820	443	45%	3600	375	12.5	582	830	30%	0.4
10-12-9-2	1014	435	56%	3600	393	31.9	698	911	23%	0.3
10-14-9-2	1125	415	62%	3600	409	68.3	803	1041	23%	0.1
20-12-19-2	-	-	-	3600	651	195.4	671	1481	55%	2.6
20-16-19-2	-	-	-	3600	655	237.7	898	1644	45%	3.2
20-20-19-2	-	-	-	3600	674	322.6	1125	1813	38%	2.3
20-24-19-2	-	-	-	3600	720	453.4	1367	2013	32%	0.9
20-28-19-2	-	-	-	3600	828	1697.9	1600	2186	27%	0.7
40-24-39-2	-	-	-	3600	1184	3600	1352	2751	51%	2.0
40-32-39-2	-	-	-	3600	1176	3600	1768	3405	48%	9.7
40-40-39-2	-	-	-	3600	1021	3600	2182	3746	42%	10.3
40-48-39-2	-	-	-	3600	1280	3600	2623	4131	36%	14.3
40-56-39-2	-	-	-	3600	-	3600	3066	4465	31%	17.1
60-36-59-2	-	-	-	3600	-	3600	1977	4631	57%	19.3
60-48-59-2	-	-	-	3600	-	3600	2675	5212	48%	25.9
60-60-59-2	-	-	-	3600	-	3600	3357	5776	42%	33.0
60-72-59-2	-	-	-	3600	-	3600	3953	6318	37%	47.8
60-84-59-2	-	-	-	3600	-	3600	4623	6950	33%	13.2

Tabela 5.3: Resultados das heurísticas. Os valores apresentados na coluna Heurísticas representam o número de vezes que as heurísticas obtiverem o melhor limite superior da iteração.

Código da Instância	Heurísticas				Total de vezes que as heurísticas foram computadas
	Heurística 1	Heurística 2	Heurística 3	Heurística 4	
5-3-4-1	3	5	2	5	15
5-4-4-1	5	5	3	3	16
5-5-4-1	4	3	3	2	12
5-6-4-1	3	2	2	1	8
5-7-4-1	2	1	1	1	5
10-6-9-1	1	3	1	2	7
10-8-9-1	1	1	1	1	4
10-10-9-1	1	1	1	1	4
10-12-9-1	1	1	1	1	4
10-14-9-1	1	1	0	1	3
20-12-19-1	1	1	0	1	3
20-16-19-1	1	1	0	1	3
20-20-19-1	1	1	0	1	3
20-24-19-1	1	1	0	1	3
20-28-19-1	1	1	0	1	3
40-24-39-1	1	1	0	1	3
40-32-39-1	1	1	0	1	3
40-40-39-1	1	1	0	1	3
40-48-39-1	1	1	0	1	3
40-56-39-1	1	1	0	1	3
60-36-59-1	1	1	0	1	3
60-48-59-1	1	1	1	1	4
60-60-59-1	1	1	0	1	3
60-72-59-1	1	1	0	1	3
60-84-59-1	1	1	0	1	3
5-3-4-2	1	1	1	1	4
5-4-4-2	1	1	1	1	4
5-5-4-2	1	1	1	1	4
5-6-4-2	1	0	1	0	2
5-7-4-2	1	1	0	1	3
10-6-9-2	1	1	0	1	3
10-8-9-2	1	1	0	1	3
10-10-9-2	1	1	1	1	4
10-12-9-2	1	1	1	1	4
10-14-9-2	1	1	0	1	3
20-12-19-2	1	1	0	1	3
20-16-19-2	1	1	0	1	3
20-20-19-2	1	1	0	1	3
20-24-19-2	1	1	0	1	3
20-28-19-2	1	1	0	1	3
40-24-39-2	1	1	0	1	3
40-32-39-2	1	1	0	1	3
40-40-39-2	1	1	0	1	3
40-48-39-2	1	1	0	1	3
40-56-39-2	1	1	0	1	3
60-36-59-2	1	1	0	1	3
60-48-59-2	1	1	1	1	4
60-60-59-2	1	1	0	1	3
60-72-59-2	1	1	0	1	3
60-84-59-2	1	1	0	1	3

Capítulo 6

Conclusões e perspectivas

Neste trabalho, um modelo matemático para o problema de sequenciamento de caminhões em um centro de *crossdocking* com duas máquinas foi analisado. Implementamos a relaxação lagrageana através do algoritmo do volume e calibramos os parâmetros do algoritmo através do Método SPOT. Demonstramos que a solução ótima dos subproblemas lagrangeanos pode ser obtida de acordo com a regra WSPT no caso do subproblema X, e através da regra WSPT-TRD no caso do subproblema Y, provando que os subproblemas são polinomiais. Além disso, quatro heurísticas construtivas polinomiais foram propostas a fim de encontrar um limite superior para o problema.

O desempenho do modelo mostrou-se dependente do número de *jobs* e do tempo de processamento. A formulação matemática aqui considerada, por empregar indexação no tempo, apresentou dificuldades para resolver problemas de maiores dimensões. Os resultados apontaram que a relaxação linear apresentou dificuldade de resolução nas instâncias de grande porte, consumindo muito tempo. Já a Relaxação Lagrangeana se mostrou bastante eficiente uma vez que obteve limites mais próximos dos valores ótimos em tempos computacionais reduzidos. O resultado da relaxação lagrangeana superou o da relaxação linear graças a redução no horizonte de tempo durante a execução do algoritmo do volume. O horizonte de tempo é reduzido a cada iteração com melhora do limite superior. Ao realizar essa redução do horizonte de tempo (T) a relaxação lagrageana obteve melhores limites inferiores que os obtidos pela relaxação linear, mesmo sendo os subproblemas polinomiais.

As heurísticas polinomiais implementadas apresentaram resultados bastante satisfatórios, não ultrapassando 2 segundos para as instâncias avaliadas. Duas das quatro heurísticas analisadas, a heurística 1 e a heurística 2, tiveram melhor desempenho na obtenção de um limite supeior, seguida pela heurística 4 que também apresentou resultados interessantes.

Diante dos fatos expostos conclui-se que a utilização da técnica de Relaxação Lagrangeana através do algoritmo do volume é uma forma eficiente para resolver o problema de sequenciamento de caminhões em um centro de *crossdocking*. Percebemos a influência da redução do horizonte de tempo na performance do algoritmo impactando diretamente na melhoria dos limites encontrados e na diminuição do tempo computacional. Além disso, o método SPOT se mostrou uma metodologia eficiente na calibração dos parâmetros do algoritmo.

Com o objetivo de buscar formas de melhorar o desempenho da relaxação Lagrangeana, propõe-se como trabalhos futuros a aplicação de uma metaheurística híbrida que utilize a relaxação Lagrangeana integrada com heurísticas para resolver o problema de modo a obter boas soluções em menor tempo computacional e com garantia de performance. Considerando-se as informações sobre os multiplicadores de Lagrange, esta metaheurística utilizará as informações destes multiplicadores para construir e perturbar soluções viáveis.

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

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

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

[7] Bartholdi, J., and Gue, K. The Best Shape for Crossdock. *Transportation Science 38 n.2* (2004), 235–244.

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

[9] 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.

[10] BookBinder, J., and Gümüs, M. Cross-docking and its implications in Location Distribution Systems. *Journal of Bussiness Logistics 5* (2004), 64–79.

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

[12] 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.

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

[14] Breiman, L. Random forests. *Machine Learning. 45* (2001), 5–32.

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

[16] 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.

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

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

[19] Fukuda, E. H. Algoritmo de volume e otimização não diferenciável. Master's thesis, USP, 2007.

[20] Gue, K. R. The Effects of Trailer Scheduling on the Layout of Freight Terminals. *Transportation Science 33* (1999), 419–428.

[21] Jayaraman, V., and Ross, A. A simulated annealing methodology to distribution network design and management. *Computers & Industrial Engineering 55(1)* (2008), 629–645.

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

[23] Lee, K., Lee, Y., and Jung, J. Vehicle routing scheduling for cross-docking distribution networks with setup costs and time window constraints. *Omega 39*, 1 (2011), 64–72.

[24] Lemaréchal, C. Lagrangian relaxation, Computational Combinatorial. *Springer Verlag*.

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

[26] Lemaréchal, C. Nondifferentiable optimization. *Optimization, Hanbooks in Operations Research* (529–572).

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

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

[33] Pinedo, M. *Scheduling: Theory, Algorithms and Systems.* 2001.

[34] Rodrigues, B., Li, Y., and Lim, A. Crossdocking - JIT scheduling with time windows. *Journal of the Operational Research Society 55(1)* (2004), 1342–1351.

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

[36] Smit, S.K.; Eiben, A. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary. 1.* (2011), 19–31.

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

[38] Sung, C., and Song, S. Integrated sevice network design for a cross-docking supply chain network. *JORS 54(12)* (2003), 1283–1295.

[39] 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.

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

[41] Vanderbeck, F., and Wolsey, L. Reformulation and decomposition of integer programs. *CORE Discussion Paper.* (2009).

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

[43] 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.

Apêndice A

Sequential Parameter Optimization Toolbox (SPOT)

Para calibrar os parâmetros do algoritmo do volume utilizamos o método SPOT, devido à sua alta eficiência. SPOT é uma aplicação do *Sequential Parameter Optimization* (SPO) que é um método baseado em modelos iterativos de calibração de algoritmos. O processo de calibração é baseado nos dados dos parâmetros e sua utilidade pelo desempenho do algoritmo do volume. SPO realiza um procedimento multi-estágio em que o modelo é atualizado a cada iteração com um conjunto de novos parâmetros e novas previsões de utilidade a fim de melhorar a eficiência do algoritmo.

O objetivo do SPOT é a determinação de boas definições de parâmetros para algoritmos heurísticos. Ele fornece ferramentas estatísticas para analisar e compreender o desempenho do algoritmo. SPOT é implementado como um pacote do R, e está disponível em http://cran.r-project.org/web/packages/SPOT/index.html.

De acordo com Bartz-Beielstein [8], SPOT pode combinar diferentes técnicas estatísticas como, por exemplo, *Design of Experiments* (DOE) que consiste na realização de uma série de testes alterando os valores das variáveis de entrada com o intuito de analisar os efeitos de cada variável ou das interações entre diferentes variáveis nos resultados obtidos. Também incorpora modelos matemáticos baseados em técnicas de regressão, tais como árvores de regressão [14].

Smit e Eiben [36] classificam SPOT como um método iterativo e explicam que os modelos baseados em técnicas de regressão utilizados por este método permitem mapear valores para os resultados dos testes através de amostras de experimentos resultantes da aplicação de DOE. Com isso, o modelo tem como objetivo prever tais resultados em decorrência do comportamento causado pelas alterações realizadas

nos valores dos parâmetros. Como em cada iteração do método ocorre a atualização desse modelo, ao atingir um número máximo de testes, são apresentados os valores de parâmetros que demonstraram melhor qualidade nos testes realizados.

Conforme explica Bartz-Beielstein [8], a implementação da SPOT consiste em duas fases: (a) construção de um modelo matemático, baseado em técnicas de regressão, e (b) melhoria sequencial deste. Na primeira fase, para criação do modelo, é determinada uma população de projetos iniciais (vetores de parâmetros cujo tamanho é igual à quantidade de parâmetros a serem calibrados) dentro do espaço de busca. Métodos como DOE podem ser usados para gerar esses projetos. O algoritmo é executado várias vezes para cada vetor de parâmetros com a finalidade de estimar o efeito desses valores nos experimentos realizados. A segunda fase consiste em um ciclo com os seguintes passos:

1. Atualização do modelo, por meio dos resultados obtidos na primeira fase;
2. Geração de uma nova população de projetos (outro conjunto de vetores de parâmetros);
3. Seleção dos melhores vetores de parâmetros e execução do algoritmo evolutivo com esses valores;
4. Caso o critério de parada não seja atingido, o ciclo inicia-se novamente.

A.1 Experimentos

Trabalhos clássicos sobre SPOT definem três diferentes fases para analisar a calibração dos parâmetros. A primeira fase é a fase de aplicação, no nosso caso, o problema com duas máquinas em *flowshop*, com restrições de *crossdocking*. A função objetivo e os parâmetros do problema são definidos nessa fase. A fase do algoritmo, está relacionada com a representação do algoritmo heurístico e com os parâmetros, que são os que determinam o desempenho do algoritmo. E, finalmente, a fase de projeto, onde o método tenta encontrar boas configurações para os parâmetros.

Dessa forma, estamos diante de dois problemas de otimização: Resolução do problema e Calibração dos parâmetros. A resolução do problema inclui a fase de aplicação e o Algoritmo do Volume a fase do algoritmo, o objetivo é encontrar uma solução ótima para o problema. A calibração dos parâmetros é realizada por meio de um método de ajuste para encontrar os melhores valores para os parâmetros do algoritmo do volume. O valor calibrado é a medida de qualidade da primeira otimização, que depende da instância a ser resolvida. Utilidade é a medida de qualidade para os

parâmetros calibrados, que reflete o desempenho do algoritmo do volume para um vetor de parâmetros.

Desse modo, para a realização da calibração foram utilizadas 20 instâncias do problema, conforme apresentado na Tabela A.1. Para definir a região de interesse (ROI) do algoritmo, o tipo e os limites inferior e superior dos parâmetros do algoritmo do volume estão resumidos na Tabela A.2.

Tabela A.1: Amostra de Instâncias

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

Tabela A.2: Limites dos parâmetros

Parâmetro	Limite Inferior	Limite Superior	Tipo
π	0.0005	0.0100	FLOAT
MaxWaste	5	30	INT
factor	0.1	1.0	FLOAT
α_{max}	0.20	0.95	FLOAT
st	0.6	1.8	FLOAT
α	0.05	0.90	FLOAT
amarela	0.20	0.95	FLOAT
verde	1.0	2.0	FLOAT

A.2 Resultados dos Experimentos

O algoritmo SPOT, implementado no pacote do R, está ligado com o algoritmo do volume implementado em C++ com o auxílio de um *callString*, tal como apresentado abaixo. O computador utilizado nos testes foi um Intel (R) Xeon (R) CPU X5690 @ 3.47GHz com 24 processadores, 132 GB de RAM, e sistema operacional Ubuntu Linux.

$callString \longleftarrow paste(\text{“./RLg ”}, \pi, MaxWaste, factor, \alpha_{max}, st, \alpha, amarela, verde)$
$call \longleftarrow system(callString, intern = TRUE)$
$\#Resultados$
$y = read.table(\text{“Resultado.txt”})$

A Tabela A.3 apresenta os quatro melhores resultados encontrados pelo método SPOT para o algoritmo do volume. A primeira coluna indica a iteração do SPOT, a segunda apresenta o vetor de parâmetros e a terceira o valor da solução obtida para as instâncias testadas.

O melhor vetor de parâmetro para o algoritmo do volume é π=0.00679, MaxWaste=24, factor=0.87753, α_{max}=0.30337, st=1.41179, α=0.08300, amarela=0.48159 e verde=1.56487.

Tabela A.3: Quatro melhores resultados encontrados pelo método SPOT

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