UNIVERSIDADE DE SÃO PAULO

ESCOLA POLITÉCNICA

FELIPE KESROUANI LEMOS

PROBLEMA DE PROGRAMAÇÃO DE UMA OPERAÇÃO DE EMPACOTAMENTO
NÃO-GUILHOTINADO EM AMBIENTE DE MÁQUINA ÚNICA, MINIMIZANDO
CUSTOS DE MATÉRIA-PRIMA E DESVIO DE DATAS: FORMULAÇÃO E SOLUÇÃO
HEURÍSTICA

São Paulo

2013

FELIPE KESROUANI LEMOS

PROBLEMA DE PROGRAMAÇÃO DE UMA OPERAÇÃO DE EMPACOTAMENTO
NÃO-GUILHOTINADO EM AMBIENTE DE MÁQUINA ÚNICA, MINIMIZANDO
CUSTOS DE MATÉRIA-PRIMA E DESVIO DE DATAS: FORMULAÇÃO E SOLUÇÃO
HEURÍSTICA

Dissertação apresentada à Escola Politécnica
da Universidade de São Paulo para obtenção
do título de Mestre em Engenharia de
Produção.

Área de Concentração:
Engenharia de Produção

Orientador: Prof. Dr. Miguel Cezar Santoro

São Paulo

2013

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

FICHA CATALOGRÁFICA

Lemos, Felipe Kesrouani

Problema de programação de uma operação de empacotamento não-guilhotinado em ambiente de máquina única, minimizando custos de matéria-prima e desvio de datas: formação e solução heurística / F.K. Lemos. -- São Paulo, 2013. 174 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Produção.

1.Programação da produção 2.Corte 3.Empacotamento I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Produção II.t.

Nome: LEMOS, Felipe Kesrouani

Título: Problema de programação de uma operação de empacotamento não-guilhotinado em ambiente de máquina única, minimizando custos de matéria-prima e desvio de datas: formulação e solução heurística.

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia de Produção.

Aprovado em:

Banca Examinadora

Prof. Dr. _____ Instituição: _____

Julgamento: _____ Assinatura: _____

Prof. Dr. _____ Instituição: _____

Julgamento: _____ Assinatura: _____

Prof. Dr. _____ Instituição: _____

Julgamento: _____ Assinatura: _____

DEDICATÓRIA

Dedico este trabalho à minha mãe, que sempre foi a primeira a apoiar minha carreira acadêmica e, de onde está, certamente está feliz pelo fim deste passo.

AGRADECIMENTOS

Ao Prof. Dr. Miguel Cezar Santoro, meu orientador, que mais uma vez foi uma fonte inesgotável de sábias orientações, ricas discussões e muito crescimento pessoal. Lembrar das muitas horas que dispendemos juntos na realização deste trabalho sempre terá uma conotação de carinho e felicidade, e nunca de sacrifício. Também agradeço à Cibele por ter me recebido tantas vezes, até mesmo em sua casa, e por ter sido uma companhia tão agradável ao longo da execução deste trabalho.

Ao Prof. Dr. Marco Aurélio de Mesquita e ao Prof. Dr. Reinaldo Morábito Neto, que fizeram ricas contribuições na banca de Qualificação deste trabalho, lapidando minhas ideias iniciais e permitindo explorar melhor o potencial deste problema estudado.

À Prof^a. Dr^a. Linda Lee Ho, que teve a sensibilidade de acolher-me temporariamente como orientado, gesto sem o qual não seria viável sequer iniciar a trilha para o mestrado.

À Lúcia Nogueira da Silva, por ter dado todo suporte institucional às etapas deste mestrado, desde a inscrição até o depósito deste trabalho. Serei sempre grato pela paciência com todos os meus erros.

Aos meus padrinhos, tia Tânia e tio Rafael, que, em pontos distintos da minha vida, me ajudaram a assumir e dividir as responsabilidades que surgiram ao longo destes 3 anos. Estes fardos seguramente seriam incompatíveis com a execução deste trabalho se não fossem as lições e ensinamentos dos meus “segundos pais”.

Ao amigo Osni e à amiga Cris, novamente meus co-orientadores, não só nesta dissertação, nem somente em minha pós-graduação, mas também na vida como um todo. Poder ter o abraço deles ao chegar na Poli sempre será motivo de segurança e força.

À todos os meus amigos que me ajudaram. Realmente, agradeço a todos, embora seja impossível enumerá-los. Em especial, gostaria de agradecer aos meus amigos Bruno, Rafael, Gil, Gouveia e Álvaro, que em momentos distintos foram minhas companhias mais próximas nestes 3 anos, torcendo e sofrendo ao meu lado para realizar este trabalho. Também agradeço à família amiga, tio Sérgio, tia Paula, Carol e meu irmão Mateus, que por tantas vezes fizeram de sua casa a minha e me receberam como membro do lar, acompanhando cada entrega e cada atraso desta dissertação.

À minha noiva Ana, que cedeu sua casa para abrigar meus momentos de concentração para executar este trabalho; e, também, seu coração para abrigar meus momentos de fraqueza, dúvidas, medos, ansiedades e, claro, os de empolgação. Mais do que ninguém, novamente, é quem sabe do suor e sangue por trás de cada tabela, cada linha de código, e cada palavra aqui escrita. Foi quem, ao mesmo tempo, me cobrou ferozmente pelo término desta dissertação e soube entender as demoras por tudo o mais que gravitou neste período. Também agradeço à sua família, que cada vez mais – ao longo deste mestrado – se transformou em parte da minha família. Obrigado pelo acolhimento com carinho em todos os momentos e pela torcida para a finalização desta dissertação.

Finalmente, à minha família, que esteve ao meu lado para apoiar nesta trilha pelo saber. Reforço a participação de meu irmão Leonardo, que mesmo tirando sarro dos meus sábados à noite com “funções”, não me deixou fraquejar neste e nenhum outro objetivo, tendo sido o maior exemplo de superação e crescimento com quem convivi neste período. Em especial, gostaria de agradecer à minha mãe, que me viu ingressar no mestrado, mas teve que continuar sua sempre animada torcida por minha carreira acadêmica de algum lugar superior ao que estamos. Sua vibração ainda é tão presente que a sinto aqui, me abraçando e parabenizando, se emocionando com qualquer traço de evolução neste trabalho.

RESUMO

LEMOS, F. K. Problema de programação de uma operação de empacotamento não-guilhotinado em ambiente de máquina única, minimizando custos de matéria-prima e desvio de datas: formulação e solução heurística. 2013. 214 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2013.

A presente pesquisa tem como objetivo estudar a integração entre dois temas clássicos da literatura de pesquisa operacional e gestão de operações: problemas de corte e empacotamento; e problemas de programação da produção. Ainda que sejam duas áreas intensamente exploradas e pesquisadas, e, ainda, que seja uma situação facilmente encontrada em sistemas de produção reais, abordagens de ambos problemas de forma coordenada ainda carecem de maiores pesquisas. Neste trabalho é feita uma revisão de ambos temas, com foco em problemas de *bin packing* e programação em ambiente de máquina única com objetivo de minimizar soma de atrasos e adiantamentos ponderados. Uma formulação matemática linear e inteira mista é proposta para o problema, contemplando as restrições que concernem a cada um e também à sua consideração simultânea. Como se trata de um problema que une dois outros, cada um NP-hard isoladamente, um método heurístico é proposto para obter uma solução interessante em tempos computacionais bastante reduzidos. Foram obtidas propriedades físicas de definição de data ideal de programação de um conjunto de itens atribuídos a um *bin*. Também é proposto um método para geração de um limitante inferior melhorado em relação a pacotes de otimização de mercado para o problema. Ambos métodos foram testados em uma massa de dados de 1.152 instâncias, geradas para retratar cenários de diferentes datas de entrega, *setups*, custos de atraso e adiantamento em relação à matéria-prima, tamanho de itens e número de itens na instância. Os resultados mostram-se largamente superiores aos obtidos por um otimizador genérico (CPLEX), embora ainda sejam *gaps* excessivamente grandes, o que reforça a dificuldade do problema.

Palavras-chave: programação da produção (*scheduling*), corte e empacotamento (*cutting and packing*), *bin packing*.

ABSTRACT

LEMOS, F. K. Scheduling problem of a non-guillotine packing operation on single-machine environment, minimizing raw material, earliness and tardiness costs: formulation and heuristic solution. 2013. 214 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2012.

The presente research aims to explore the integration between two classic themes on operations research and operations management literature: cutting and packing problems; and production scheduling problems. Although they are intensive explored and researched areas and, besides, it's an easily found situation on real production systems, coordinated approaches of both themes still need deeper researchs. On this paper, it was done a review of both themes, focusing on bin packing problems and single-machine environment scheduling problems aiming to minimize total weighed earliness and tardiness. A mixed integer-linear mathematical formulation is proposed to the problem, including constraints referred to each problem and, also, to their simultaneous consideration. Once it's a problem that joins the other two, each one NP-hard solely, an heuristic method is proposed to obtain an interesting solution in reasonable computational times. Physical properties were identified, defining the best date to allocate a given lot of itens to be processed together. Also, a lower bound generation method is proposed, improving the one generated by optimization softwares. Both methods were tested on a 1.152 instances mass of data, generated to represent well several scenarios of different due dates, setup times, earliness and tardiness costs compared to raw material, size of items and number os items on instance. Results show largely superiority the ones obtained by an optimization pack (CPLEX), although gaps are still excessively large, fact the reinforces problem's difficulty.

Palavras-chave: production scheduling, cutting and packing, bin packing..

LISTA DE FIGURAS

Figura 1.1. Ilustração do problema de corte e empacotamento (elaboração do autor)	23
Figura 1.2. Relatório da <i>ISI Web of Knowledge</i> a respeito das publicações e citações na área de Pesquisa Operacional com as palavras-chave “cutting” ou “packing” (<i>Fonte: ISI Web of Knowledge</i> , em 10/12/2011).....	24
Figura 1.3. Relatório da <i>ISI Web of Knowledge</i> a respeito das publicações e citações na área de Pesquisa Operacional com a palavra-chave “scheduling” (<i>Fonte: ISI Web of Knowledge</i> , em 10/12/2011).....	24
Figura 1.4. Ilustração do problema de pesquisa (elaboração do autor)	25
Figura 1.5 – Exemplo de operação de corte típica da indústria de papel e metalúrgica (<i>Fonte: adaptado de Poldi, 2007</i>)	26
Figura 1.6 - Exemplo de operação de corte típica da indústria de vidro (<i>Fonte: adaptado de http://www.portalsaofrancisco.com.br</i> , em 11/12/2011)	27
Figura 1.7 - Exemplo de operação de corte típica da indústria de vidro (<i>Fonte: adaptado de GRAMANI, 2001</i>)	27
Figura 1.8 - Exemplo de operação de corte com jato d’água, sem restrição de corte guilhotinado ou ortogonal (<i>Fonte: adaptado de http://www.manutencaoesuprimentos.com.br</i> , em 11/12/2011).....	28
Figura 1.9 – Ilustração de uma situação típica de uma operação de corte (elaboração do autor)	30
Figura 1.10 – Ilustração das decisões envolvidas na programação de uma operação de corte (elaboração do autor)	32
Figura 2.1 – Hierarquização dos problemas de corte e empacotamento (<i>Fonte: Wäscher, Haubner e Schumann (2007)</i>).....	39
Figura 2.2 – Exemplos de padrões de corte com restrições de guilhotinagem (<i>Fonte: Arenales e Morábito, 1995</i>)	39
Figura 2.3 – Exemplos de padrões de corte não-guilhotinados de primeira ordem (<i>Fonte: Arenales e Morábito, 1995</i>).....	40

Figura 2.4 – Ilustração da função identicamente viável exemplificada (Fonte: Carrier, Clautiaux e Moukrim, 2007).....	49
Figura 2.5 – Ilustração da representação de uma solução para o problema de <i>bin packing</i> bidimensional por teoria dos grafos (Fonte: Fekete e Schepers, 2000a)	50
Figura 2.6 – Ilustração dos processos de alocação, <i>First-Fit Decreasing Height</i> e <i>Best-Fit Decreasing Height</i> (Fonte: Lodi, Martello e Vigo, 2002)	51
Figura 2.7 – Ilustração dos processos de alocação de níveis nos <i>bins</i> (Fonte: Lodi, Martello e Vigo, 2002).....	52
Figura 2.8 – Ilustração dos processos de alocação de itens no <i>Floor Ceiling</i> (Fonte: Lodi, Martello e Vigo, 2002).....	52
Figura 2.9 – Ilustração do processo utilizado para o raciocínio de programação dinâmica (Fonte: Liu, Chu e Wang, 2011).....	53
Figura 2.10 - Ilustração do ambiente de máquina única ($\alpha = 1$) (Fonte: elaboração do autor)	55
Figura 2.11 - Ilustração do ambiente de máquinas em paralelo ($\alpha = P2$) (Fonte: elaboração do autor)	55
Figura 2.12 – Ilustração do ambiente <i>flow shop</i> ($\alpha = F3$) (Fonte: elaboração do autor) ...	56
Figura 2.13 - Ilustração do ambiente <i>flow shop</i> flexível ($\alpha = FF3$) (Fonte: elaboração do autor)	56
Figura 2.14 - Ilustração do ambiente <i>job shop</i> ($\alpha = J6$) (Fonte: elaboração do autor)	56
Figura 2.15 – Ilustração do ambiente <i>job shop</i> flexível ($\alpha = JF6$) (Fonte: elaboração do autor)	57
Figura 2.16 – Classificação hierarquizada dos modelos de solução para o problema de <i>job shop scheduling</i> (Fonte: Pachecho e Santoro, 1999)	60
Figura 3.1 – Exemplo de conjunto de dados para o problema (exceto custos)	85
Figura 3.2 – Resultados de programações de acordo com os custos de matéria-prima, adiantamento e atraso.....	86
Figura 4.1 – Exemplo de instância com os prazos ilustrados graficamente em uma linha do tempo (Fonte: elaboração pelo autor)	88

Figura 4.2 – Ilustração de uma ideia intuitiva de empacotamento (Fonte: elaboração do autor)	89
Figura 4.3 – Ilustração da função $P(t)$, que tem formato linear por partes, concavidade para cima, tendo um ponto de mínimo, antes do qual é sempre decrescente e depois do qual é sempre crescente (Fonte: elaboração do autor)	92
Figura 4.4 – Passos 1 a 3 do algoritmo da heurística construtiva	96
Figura 4.5 – Ilustração do passo 4 do algoritmo e suas três ramificações	97
Figura 4.6 – Ilustração do passo 6 do algoritmo	98
Figura 4.7 – Ilustração da ramificação de exploração do processo de geração do <i>lower bound</i> (Fonte: elaboração do autor)	102

LISTA DE TABELAS

Tabela 2-1 – Lista de regras de despacho simpels (Fonte: Pinedo, 2008).....	66
Tabela 5-1 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “pequenos” (classe 10).....	107
Tabela 5-2 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “estreitos” (classe 8).....	108
Tabela 5-3 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “largos” (classe 7).....	109
Tabela 5-4 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “grandes” (classe 9).....	110
Tabela 5-5 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “pequenos” (classe 10).....	111
Tabela 5-6 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “estreitos” (classe 8).....	112
Tabela 5-7 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “largos” (classe 7).....	113
Tabela 5-8 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “grandes” (classe 9).....	114
Tabela 5-9 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “pequenos” (classe 10).....	115
Tabela 5-10 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “estreitos” (classe 8).....	116
Tabela 5-11 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “largos” (classe 7).....	117
Tabela 5-12 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “grandes” (classe 9).....	118
Tabela 6-1 – <i>Gaps</i> obtidos nos diferentes cenários experimentais.....	120

Tabela 6-2 – Comparação percentual entre os métodos propostos no capítulo 4 e os resultados obtidos no CPLEX	122
Tabela 6-3 – Porcentagem de instâncias em que os métodos propostos foram superiores ao CPLEX	123
Tabela 6-4 – Comparação percentual dos limitantes superiores quebrados por classe e porte, com destaque para a classe 9 (“itens grandes”).....	124
Tabela 6-5 – Comparação percentual dos <i>upper bounds</i> destacando os custos unitários de atraso e adiantamento.....	128
Tabela 6-6 – Comparação percentual dos <i>lower bounds</i> destacando os custos unitários de atraso e adiantamento.....	129
Tabela E-1 – Sumário de estatísticas descritivas para a ANOVA dos <i>gaps</i> obtidos pelo CPLEX entre as classes de tamanhos de itens	168
Tabela E-2 – ANOVA dos <i>gaps</i> obtidos pelo CPLEX entre as classes de tamanhos de itens	168
Tabela E-3 – Sumário de estatísticas descritivas para a ANOVA dos <i>gaps</i> obtidos pelos métodos propostos entre as classes de tamanhos de itens.....	168
Tabela E-4 – ANOVA dos <i>gaps</i> obtidos pelos métodos propostos entre as classes de tamanhos de itens	168
Tabela E-5 – Teste t de Student de diferença de médias dos <i>upper bounds</i> obtidos pelo CPLEX e pela heurística construtiva proposta com 5% de significância.....	169
Tabela E-6 – Teste t de Student de diferença de médias dos <i>lower bounds</i> obtidos pelo CPLEX e pela heurística construtiva proposta com 5% de significância.....	169
Tabela E-7 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>upper bounds</i> da heurística construtiva e do CPLEX entre as classes de tamanhos de itens	169
Tabela E-8 – ANOVA das comparações percentuais dos <i>upper bounds</i> da heurística construtiva e do CPLEX entre as classes de tamanhos de itens	169
Tabela E-9 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre as classes de tamanhos de itens .	170

Tabela E-10 – ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre as classes de tamanhos de itens.....	170
Tabela E-11 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre os portes de instâncias (número de itens)	170
Tabela E-12 – ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre os portes de instâncias (número de itens)	170
Tabela E-13 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre os portes de instâncias (número de itens)	171
Tabela E-14 – ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre os portes de instâncias (número de itens)	171
Tabela E-15 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β).....	171
Tabela E-16 – ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)	171
Tabela E-17 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β).....	172
Tabela E-18 – ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)	172
Tabela E- 19 – Teste t de Student de diferença de médias da comparação percentual dos <i>lower bounds</i> para instâncias de custos de adiantamento (α) altos e baixos com 5% de significância	172
Tabela E-20 – Teste t de Student de diferença de médias da comparação percentual dos <i>upper bounds</i> para instâncias de <i>setups</i> curtos e longos com 5% de significância	173
Tabela E-21 – Teste t de Student de diferença de médias da comparação percentual dos <i>lower bounds</i> para instâncias de <i>setups</i> curtos e longos com 5% de significância	173

Tabela E-22 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre diferentes prazos	173
Tabela E-23 – ANOVA das comparações percentuais dos <i>upper bounds</i> do método proposto e do CPLEX entre diferentes prazos	173
Tabela E-24 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre diferentes prazos.....	174
Tabela E-25 – ANOVA das comparações percentuais dos <i>lower bounds</i> do método proposto e do CPLEX entre diferentes prazos	174

LISTA DE GRÁFICOS

Gráfico 6-1 – Comparação entre <i>upper bounds</i> nos diferentes cenários experimentais, destacando o número de itens.....	126
Gráfico 6-2 – Comparação entre <i>lower bounds</i> nos diferentes cenários experimentais, destacando o número de itens.....	127
Gráfico 6-3 – Comparação percentual do <i>upper bound</i> entre instâncias com <i>setups</i> curtos e longos sob a ótica de outros fatores experimentais	130
Gráfico 6-4 – Comparação percentual do <i>lower bound</i> entre instâncias com <i>setups</i> curtos e longos sob a ótica de outros fatores experimentais	130
Gráfico 6-5 – Comparação percentual do <i>upper bound</i> entre instâncias com datas de entrega próximas, distantes ou variadas sob a ótica de outros fatores experimentais	131
Gráfico 6-6 – Comparação percentual do <i>lower bound</i> entre instâncias com datas de entrega próximas, distantes ou variadas sob a ótica de outros fatores experimentais	132

LISTA DE QUADROS

Quadro 2.1 – Formulação discretizada do 2DBPP (Beasley, 1985).....	43
Quadro 2.2 – Formulação contínua com restrições disjuntivas para o 3DBPP (Hifi et al., 2010)	44
Quadro 2.4 – Formulação para escolha de padrões de corte para resolução por geração de colunas (Morábito, 1994)	45
Quadro 3.1 – Cenários de S_{ij} e S_{ji} e suas consequências na formulação.....	82
Quadro 3.2 – Exemplo de computação de b_i em que em um <i>bin</i> são alocados os itens 2 e 4 e outro, programado posteriormente, os itens 1, 3 e 5.....	84
Quadro 4.1 – Algoritmo da heurística construtiva proposta	95
Quadro 4.2 – Algoritmo da heurística de geração do <i>lower bound</i> do problema.....	100

SUMÁRIO

1. INTRODUÇÃO	22
1.1. Motivação	22
1.2. Problema da programação de operações de corte e empacotamento	25
1.2.1. Exemplos de operações de corte na indústria	26
1.3. Abrangência do trabalho e objeto de estudo.....	28
1.4. Descrição física do problema de empacotamento abordado.....	30
1.5. Objetivo, justificativa e método	33
1.6. Organização do trabalho.....	34
2. REVISÃO BIBLIOGRÁFICA	36
2.1. Corte e empacotamento	36
2.1.2. <i>Bin packing</i> bidimensional orientado com corte livre (2DBP OIF).....	41
2.2. Programação da produção (<i>scheduling</i>)	53
2.2.1. Tipologia	54
2.2.2. Métodos de solução	59
2.2.3. Programação em máquina única com objetivo de minimizar soma das penalidades por atraso e adiantamento ($1 \alpha Ej + \beta jTj$).....	61
2.3. Abordagens conjuntas entre programação e operações de corte	67
2.3.1. Problema das pilhas abertas	67
2.3.2. Dimensionamento de lotes	68
2.3.3. Atendimento de datas	68
2.3.4. Outras abordagens	69
2.4. Considerações finais.....	69
3. MODELAGEM MATEMÁTICA PARA O PROBLEMA.....	70
4.1. Modelagens específicas	70

4.1.1. Modelo para o problema de <i>bin packing</i>	70
4.1.2. Modelo para o problema <i>scheduling</i>	74
4.2. Modelagens para o problema de programação de operações de corte	76
4. MÉTODOS HEURÍSTICOS	87
4.1. Heurística construtiva.....	87
4.1.1. Intuição e bases teóricas	88
4.1.2. Algoritmo	95
4.2. Heurística para obtenção do <i>lower bound</i>	98
5. EXPERIMENTOS COMPUTACIONAIS	103
5.1. Geração de instâncias	103
5.2. Procedimento experimental	105
5.3. Resultados.....	107
6. DISCUSSÃO	119
6.1. Formulação e dificuldade do problema	119
6.2. Comparação entre os métodos propostos e o <i>software</i> de mercado (CPLEX)	121
6.2.1. Classes (tamanho dos itens)	124
6.2.2. Porte da instância (número de itens).....	125
6.2.3. Custos relativos de adiantamento, atraso e matéria-prima	127
6.2.4. Setup	129
6.2.5. Datas de entrega	131
7. CONCLUSÕES E EXTENSÕES	133
APÊNDICE A – Código fonte para geração das instâncias em VBA	143
APÊNDICE B – Código fonte para os experimentos no CPLEX através do IBM ILOG CPLEX Optimization Studio.....	147
B.1. Script (<i>Script.mod</i>).....	147
B.2. Modelo (<i>Modelo.mod</i>).....	148

<i>B.3. Exemplo de arquivo de dados (l.dat)</i>	150
APÊNDICE C – Código fonte para a heurística construtiva em VBA.....	151
Apêndice D – Código fonte para a geração de <i>lower bound</i> segundo o método proposto em VBA	154
APÊNDICE E – Detalhamento de testes estatísticos.....	168

1. INTRODUÇÃO

O intuito deste primeiro capítulo é apresentar o presente trabalho. São expostos aqui a sua motivação inicial, lastreada tanto na relevância do tema para a indústria quanto para a literatura; o enunciado formal do problema, bem como uma explanação mais longa de suas características, simplificações e aplicações práticas; são apontados os seus objetivos e sua justificativa, que determinam sua abrangência através da delimitação do recorte utilizado; e, por fim, é apresentada a estruturação dos demais capítulos, para orientar o leitor nas seções posteriores.

1.1. Motivação

Os problemas de programação da produção (*scheduling*) são vastamente conhecidos na literatura. Trata-se do processo de tomada de decisão associado à alocação de recursos a tarefas no tempo, com o intento de otimizar um ou mais objetivos (PINEDO, 2008).

A pesquisa na área formalmente se inicia com o trabalho de Kelley e Walker (1959) com o algoritmo de determinação de programação de atividades nos arcos para redes de precedência, embora já exista a preocupação inerente a esta questão desde o início da administração científica com os gráficos de Gantt na década de 1920.

Com o avanço da computação, o campo é amplamente explorado nas mais diversas variantes de ambientes de produção, objetivos de otimização e restrições inerentes, sejam elas técnicas ou mercadológicas. Entretanto, no *survey* feito por Craighead e LaForge (2003) em indústrias americanas constatou-se uma utilização de ferramentas de programação finita apenas em 25% delas. Isso mostra que, embora avançado, o espaço para melhorias ao encontro de situações reais ainda é grande, como reforçam Gupta e Stafford (2006) no caso específico de ambiente *flowshop*.

Outra área amplamente explorada na literatura são os chamados problemas de corte e empacotamento. Este campo estuda a alocação de um conjunto de itens menores em objetos maiores obedecendo a restrições de não-sobreposição e respeito aos limites dos objetos, com o intuito de otimizar seu aproveitamento (WASCHER et al., 2007). O interesse nesta área,

segundo Bischoff e Wascher (1995), pode ser atribuído à (i) aplicabilidade, (ii) diversidade de problemas reais e (iii) sua complexidade.

Figura 1.1. Ilustração do problema de corte e empacotamento (elaboração do autor)

Desde o trabalho seminal de Kantorovich (1960 apud Ortmann et al., 2010), verifica-se uma grande ramificação do problema original, com diversas variantes em termos de dimensões, homogeneidade e formato dos itens, objetivo de maximizar valor de itens ou minimizar uso de objetos (WASCHER et al., 2007).

Como ressalta Allen et al. (2011), a gama de aplicações industriais dos problemas de corte e empacotamento é notável: indústrias de papel, aço, alumínio, vidro e qualquer outra natureza que lida com chapas ou placas como insumo. Além disto, aplicações não imediatamente relacionadas com problemas de corte, recaem na mesma formulação teórica. Exemplos de situações de empacotamento são mencionados por Dyckhoff (1990), como o problema de balanceamento de linhas, formulação de orçamentos, *layout* de unidades de produção ou mesmo de publicações de imprensa, esta última mencionada por Downslan e Downslan (1995).

Ambos campos – programação da produção e corte e empacotamento – apresentam amplo desenvolvimento nas últimas décadas e a quantidade de publicações mantém-se representativa e crescente nos últimos anos (ver Figuras 1.3 e 1.4).

Figura 1.2. Relatório da ISI Web of Knowledge a respeito das publicações e citações na área de Pesquisa Operacional com as palavras-chave “cutting” ou “packing” (Fonte: ISI Web of Knowledge, em 10/12/2011)

Figura 1.3. Relatório da ISI Web of Knowledge a respeito das publicações e citações na área de Pesquisa Operacional com a palavra-chave “scheduling” (Fonte: ISI Web of Knowledge, em 10/12/2011)

Em que pese a grande relevância de ambos temas e seu interesse tanto por pesquisadores quanto pela indústria, poucas são as abordagens conjuntas deste tipo de problema. Em contraponto a esta lacuna da literatura, os sistemas de produção reais de operações de corte apresentam objetivos das duas áreas como preocupações inerentes: (i) atendimento de datas dos itens cortados; e (ii) bom aproveitamento das matérias-primas. Casos interessantes retratados na literatura são a indústria papelreira (POLTRONIERI, 2006), a indústria moveleira (GRAMANI, 2001) e a indústria aeronáutica (LEMOs e SANTORO, 2010).

Importante observar que, ainda que tais trabalhos retratam sistemas de produção cujo problema de corte sob a ótica da preocupação multi-período, seus focos não estão na

programação das tarefas (*scheduling*). Os dois primeiros voltam-se ao dimensionamento de lotes, enquanto o último apenas ao sequenciamento das tarefas.

Desta forma, a principal motivação para o presente trabalho, como ilustrado na Figura 1.5, é explorar a programação de operações de corte. Trata-se de um problema que (i) situa-se na junção de dois problemas clássicos da literatura, como apresentado anteriormente; (ii) apresenta grande relevância para a indústria, já que existem plantas com operações desta natureza, inseridas em sistemas cujo objetivo não é só otimizar o uso de matérias-primas, mas, também, o atendimento de datas; e (iii) apresenta complexidade suficiente para fomentar e suscitar extensões e desdobramentos.

Figura 1.4. Ilustração do problema de pesquisa (elaboração do autor)

1.2. Problema da programação de operações de corte e empacotamento

Seja um conjunto de n itens retangulares a serem programados em uma operação de corte ou empacotamento ($i = 1, 2, \dots, n$) com tempo de processamento individuais p_i , data de entrega d_i , penalidade por unidade de tempo de término em adiantamento α_i , penalidade por unidade de tempo de término em atraso β_i , dimensões l_i (comprimento) e w_i (largura), não podendo ser rotacionados. Seja um conjunto de objetos (*bins*) idênticos de comprimento L , largura W , custo θ e tempo de preparação s_t , nos quais os n itens mencionados serão alocados para sofrerem uma operação industrial formando um lote, sendo sua alocação não-guilhotinada. O objetivo do problema é agrupar os n itens em lotes a serem processados de acordo com a capacidade destes *bins*, que posteriormente serão programados de forma a minimizar a soma das penalidades por adiantamentos, atrasos dos itens, além do custo do uso de matéria-prima.

1.2.1. Exemplos de operações de corte na indústria

Operações de corte ocorrem nos mais distintos sistemas de manufatura em uma grande variedade de formatos.

Na indústria de papel e metalúrgica, que são exemplos clássicos deste problema, grandes unidades chamadas de “jumbos” ou “rolos jumbos” são produzidas e estocadas, tendo versatilidade para gerar produtos de diversos tamanhos em termos do comprimento axial. Quando de sua necessidade em um determinado comprimento exigido por um produto (chapa metálica ou folha de papel), estes rolos são cortados e transformados em bobinas de comprimento menor, que continuarão posteriormente no processo até atingirem o formato de produto final.

Figura 1.5 – Exemplo de operação de corte típica da indústria de papel e metalúrgica
(Fonte: adaptado de Poldi, 2007)

Este processo, ilustrado na Figura 1.6, é um exemplo de operação de corte unidimensional, em que uma unidade maior é subdividida em unidades menores através de um corte que delimita um comprimento na direção axial do jumbo. As bobinas são cortadas de acordo com as dimensões demandadas e procura-se diminuir o desperdício (sobras), especialmente quando se trata de produtos de alto valor agregado.

Outro exemplo conhecido de operação de corte unidimensional é a indústria de vidro, como mencionado por Yuen (1991). A manufatura do vidro *float* é um processo contínuo que termina com o recorte unidimensional de chapas com tamanhos pré-determinados que atendam ao dimensionamento da demanda. A Figura 1.7 ilustra esse processo, ressaltando aqui que o corte parte de uma grande unidade contínua (dimensão infinita) e busca cortar de maneira a utilizar o mínimo de material para confeccionar seus produtos.

Figura 1.6 - Exemplo de operação de corte típica da indústria de vidro
(Fonte: adaptado de <http://www.portalsaofrancisco.com.br>, em 11/12/2011)

A indústria moveleira também apresenta tal questão, no início de seu processo, porém na variante bidimensional. Grandes placas de madeira, em geral mais padronizadas, são insumos para o processo de corte que dá origem a uma gama maior de peças que servirão para a montagem de módulos moveleiros. Neste caso, o cuidado com o aproveitamento do material é feito em duas dimensões, tendo, na maioria dos casos, a restrição de corte guilhotinado por ser feito com equipamentos de serra.

Figura 1.7 - Exemplo de operação de corte típica da indústria de vidro
(Fonte: adaptado de GRAMANI, 2001)

O mesmo tipo de problema se repete em indústrias metal-mecânicas, notoriamente em indústrias automobilísticas, autopeças e indústria aeronáutica. Esta última, estudada por Santoro e Lemos (2010), possui como componente adicional o alto valor agregado de seu material (placas de titânio ou alumínio aeronáutico), ao mesmo tempo de uma grande pressão de suas montagens por atendimento de datas. Nestas indústrias, além do processo semelhante à indústria moveleira, é comum também haver processos de corte por fresa (*nesting*), punção, jato d'água ou plasma. Nestes, por vezes, não existe a necessidade de fazer cortes guilhotinados e até mesmo formatos irregulares são possivelmente feitos.

Figura 1.8 - Exemplo de operação de corte com jato d'água, sem restrição de corte guilhotinado ou ortogonal (Fonte: adaptado de <http://www.manutencaoesuprimentos.com.br>, em 11/12/2011)

Muitos outros exemplos poderiam ser citados. Vamos nos ater aqui nesta restrita exposição que tem o intuito de contextualizar estas operações em sistemas de produção, vínculo este que dá sentido a este trabalho.

Todas estas operações, quando vistas de maneira isolada, têm em comum o objetivo imediato de aproveitamento de matéria-prima. As sobras são desperdícios inerentes ao processo, que aumentam o custo final do produto, geram resíduos e não agregam qualquer valor exceto seu baixo preço como sucata.

Por outro lado, como ressaltado em todos os exemplos, estes são todos processos que se inserem em um sistema maior, com objetivos de produção comuns a qualquer empresa: velocidade de fluxo (e, por consequência, menor estoque em processo), atendimento aos prazos estipulados (evitando atrasos para manter um bom nível de serviço e adiantamentos para ter estoques controlados) e boa utilização de seus equipamentos. Por essa visão, as operações em lote apresentam distinções em relação a seus pares unitários, já que os objetivos de programação podem conflitar ou acentuar os anteriores de aproveitamento de matéria-prima.

1.3. Abrangência do trabalho e objeto de estudo

Como visto, diversas são as características técnicas que permeiam o problema. Para efeito desta pesquisa, cujo principal objetivo é explorar a integração dos elementos de programação e operações de corte e empacotamento, optou-se por restringir seu escopo a um determinado tipo de operação.

São abordadas neste trabalho as operações de empacotamento em objetos (*bin packing*) bidimensionais de tamanho único, com itens de dimensões variadas, retangulares, orientados (rotações não permitidas), sem restrição de guilhotinagem.

Importante ressaltar que algumas simplificações são feitas a título de modelagem:

- O *setup* dos lotes (st) é considerado um valor fixo e próprio do *bin*. Tal escolha se deu pelo fato que, na maioria dos processos em que ocorre uma preparação prévia da máquina, o tempo gasto é função da dificuldade de posicionar este grande objeto na máquina a ser processada, tendo relação com seu peso, a precisão de posicionamento e regulagem da programação de corte. Isto particularmente tem um distanciamento da realidade em operações manuais, em que a preparação pode também ter relação com o número de itens. Entretanto, a opção por um tempo fixo parece retratar uma gama grande de sistemas de produção, especialmente os que requerem um aprofundamento maior na precisão da programação de suas atividades de corte;
- O tempo de processamento dos lotes é considerado como uma soma de tempos de processamento individuais dos itens (p_i). Isto não é precisamente aderente à realidade, na medida em que a quantidade de interfaces entre objetos retangulares influencia na economia de tempo de corte, além de que a disposição dos itens muda o tempo total de processamento (por exemplo, itens na borda do *bin* não tem uma de suas arestas não cortadas). Embora esta seja uma simplificação da realidade implícita ao problema, em um limite com grande quantidade de itens, pode-se considerar que estes efeitos são pequenos comparados ao tempo total de processamento de um lote. Além disto, a modelagem de tempos de processamento em função do posicionamento dos itens traria complexidade matematicamente adicional a um problema que já se mostrou muito demandante de esforço computacional;
- A granularidade das datas (d_i), tratadas individualmente, não se trata de uma simplificação (na verdade, é uma generalização), entretanto é um elemento importante

a ser destacado em sua forma de tratamento neste trabalho. Isto porque, ao optar por estabelecer prazos na mesma unidade de programação das máquinas, um custo alto em termos computacionais é adicionado. Em muitos casos reais, seria possível estabelecer datas em macro-períodos, já que os tempos de processamento são dados em unidades muito menores (horas ou minutos) do que as *due dates* (em dias ou semanas). Entretanto, como este estudo é uma exploração do problema e não se ancora em uma ou outra aplicação prática específica, optou-se por modelar o caso genérico, mesmo onerando o modelo computacional e inviabilizando, por exemplo, o uso de um *grid* de datas para modelar discretamente o tempo.

1.4. Descrição física do problema de empacotamento abordado

A Figura 1.10 faz uma primeira ilustração da situação de um problema de programação de uma operação de corte.

Figura 1.9 – Ilustração de uma situação típica de uma operação de corte (elaboração do autor)

Nesta ilustração, temos alguns dos **elementos** que compõem o problema:

- i. um estoque de grandes objetos idênticos de dimensões fixas e com determinado custo;
- ii. uma fila de ordens que representam demandas por pequenos itens a serem cortados. Cada uma destas ordens possui dimensões pré-determinadas; uma data

de entrega esperada, bem como um custo esperado por unidade de tempo em atraso ou adiantamento em relação a esta;

- iii. um processo de produção em que:
 - a. Várias ordens podem ser feitas ao mesmo tempo, havendo uma restrição geométrica de empacotamento dos itens nos objetos formando um padrão de corte;
 - b. Existe um tempo de preparação do objeto na máquina, ou seja, um tempo de *setup* por lote;
 - c. Existe um tempo de processamento do padrão de corte, que é função da complexidade deste.

Com este cenário em mãos, o programador tem uma série de **objetivos**, que podem ser conflitantes quanto a decisão de quê e quando fabricar:

- Quanto maior o preenchimento da área do objeto, maior a utilização do recurso matéria-prima. Por consequência, menos lotes serão feitos, sendo maior a utilização do recurso máquina (menos *setups*). Ao mesmo tempo, quanto mais peças em um padrão de corte, maior é a produtividade do sistema medida em termos de número de ordens feito por unidade de tempo (e, por consequência, menor o estoque em processo);
- Por outro lado, quanto mais próximo da data de entrega do item for o término da operação, menores as penalidade por atraso e adiantamento. Assim, quando mais agregados em lotes únicos, itens distantes em prazos tendem a sofrer maiores penalidades por desvio de datas.

Importante observar como estes objetivos potencialmente podem ser conflitantes. É fácil imaginar uma situação em que um item do final da fila de ordens encaixa-se perfeitamente em um padrão de corte que está prestes a entrar na máquina, porém isto causaria um adiantamento excessivo de seu término e, porventura pode tomar o lugar de outro item com pior encaixe no objeto (*fitting*), porém com prazos mais curtos. Por outra visão, pode-se imaginar um padrão de corte que contempla todos os itens atrasados, porém com um aproveitamento do objeto muito inferior ao que potencialmente poderia ser obtido mesclando outros itens da fila com menor prioridades.

Tais *trade-offs* permeiam e dificultam a decisão do programador que, isoladamente, já seriam bastante difíceis (só preocupar-se com aproveitamento de materiais, ou só preocupar-se com atendimento de datas).

Assim, quando refere-se a programar a produção de uma operação como esta, a **decisão** implícita ao programador, que está ilustrada na Figura 1.11 é composta por:

- Decidir quais itens devem ser agrupados entre si para formar padrões de corte;
- Decidir o instante em que tais padrões de corte devem ser cortados;
- Tomar estas decisões com o objetivo de minimizar o custo total entre uso de matéria-prima e penalidades por atraso (mal atendimento ao cliente ou restante do processo) e adiantamento (estoques).

Figura 1.10 – Ilustração das decisões envolvidas na programação de uma operação de corte (elaboração do autor)

Estas decisões não só são dificultadas pelos múltiplos objetivos conflitantes, mas também porque este problema de programação herda as **restrições** tanto temporais quanto geométricas. Ao elaborar uma solução tentativa para a programação desta operação, o decisor deve respeitar:

- a não-sobreposição entre itens geométrica, ou seja, suas áreas não podem ocupar espaços em comum no objeto;

- a pertinência do item ao objeto, ou seja, o item deve estar completamente posicionado no interior deste;
- a não-sobreposição temporal entre padrões de corte, ou seja, o início da operação de um padrão deve distar do início de seu antecessor, pelo menos, do tempo de processo deste último e do tempo de *setup* de um objeto.

1.5. Objetivo, justificativa e método

Seja o problema descrito no primeiro parágrafo do item 1.2, ancorado em suas potenciais aplicações práticas (item 1.2.1), limitado pelos recortes definidos (item 1.3) e que expressa dilemas de um processo industrial (item 1.4). Mediante este problema, o **objetivo desta dissertação é propor um modelo de otimização** para o problema de programação de operações de empacotamento não-guilhotinado, vista em ambiente de máquina única, buscando otimizar ao mesmo tempo os custos de uso de matéria-prima e desvio de datas. Além de sua formulação, também é objetivo do trabalho **desenvolver um método de solução** em tempo computacional razoável em uma **massa de instâncias** que permita retratar diversos cenários de produção.

Assim, pretende-se obter uma visão do problema integrado de atendimento de datas e aproveitamento de matéria-prima através de um modelo de programação linear inteira mista, possibilitando explorar, dentre outros pontos, a (i) complexidade do problema, (ii) os diferentes cenários de dados de entrada e seu impacto na otimização, e (iii) métodos de soluções heurísticos.

Tendo em vista a constatação do baixo uso de técnicas formais de programação na indústria, um objetivo indireto do trabalho é contribuir para o enriquecimento de técnicas disponíveis aos gestores da produção, estudando o problema de programação com as restrições técnicas da operação de corte.

Especificamente, neste trabalho, o método utilizado para explorar os objetivos apresentados é a proposição do modelo otimizante integrado de programação de operações de corte e um método de solução heurístico em tempo computacional razoável, explorando-os à luz de uma massa de dados extensiva e completa, criada com o objetivo de avaliar diferentes situações reais de tipos de itens a serem cortados e prazos para tal.

Utilizando a tipologia de metodologias quantitativas em modelagem e simulação de Morábito e Pureza (2010), esta pesquisa pode ser classificada como empírica, já que apresenta como intuito maior a adesão entre a realidade das operações de corte e o modelo proposto. Ainda pode ser considerada como empírica prescritiva, na medida em que busca apoiar o processo de decisão de programar itens a serem cortados.

Por outro lado, ao enveredar na busca de um método de solução heurístico para o problema proposto, pode-se considerar que parte do trabalho possui uma metodologia axiomática, também prescritiva. É axiomática a fim de melhorar os próprios resultados do modelo proposto (já que a sua complexidade matemática é muito alta) e prescritiva, pois segue no objetivo de melhorar o processo de tomada de decisão de programação.

Como já ressaltado anteriormente, a justificativa principal deste trabalho é explorar um tema que integra duas áreas clássicas da literatura de pesquisa operacional e gestão de operações. Sua relevância é especialmente interessante na medida em que a aplicabilidade prática das técnicas de otimização de aproveitamento de matéria-prima em operações de corte são restritas pela necessidade de obedecer a datas do restante do sistema de produção, o que provoca a necessidade de uma visão conjunta com a programação da produção.

1.6. Organização do trabalho

Este trabalho está organizado em 7 capítulos incluindo o atual, no qual o tema e o objetivo do trabalho foram delimitados, com a devida justificativa, apresentação da abordagem e método utilizados, apresentação do problema da programação de operações de corte do ponto de vista físico, contextualizando o leitor em sua lógica, suas restrições e as decisões associadas.

O capítulo 2 apresenta a revisão bibliográfica a respeito do tema, havendo a necessidade inicial de uma bifurcação entre a revisão da literatura relativa a programação da produção e relacionada a corte e empacotamento. Por fim, os trabalhos que apresentam características mistas são tratados com maior importância.

No capítulo 3 é apresentada uma proposta de modelagem matemática para o problema proposto. Uma vez que se trata de uma nova versão do problema na literatura, foi desenvolvido um modelo de programação linear inteira mista que contemple as características elencadas no capítulo 1. Para isso, utilizou-se de uma combinação adaptada de diferentes

abordagens interessantes vistas na revisão da literatura de *scheduling* e corte e empacotamento.

Como o problema deriva-se de outras duas formulações que são NP-hard, obtém-se a solução ótima apenas para números bastante restritos de itens, de forma que no capítulo 4 é proposto um método de solução para o problema. Além deste, com fins de avaliação de sua qualidade e comparação com o desempenho computacional de *softwares* de mercado no capítulo experimental, também é proposto um método de cálculo de um limitante inferior para o problema, lastreado em suas características físicas.

No capítulo 5, apresenta-se o plano de experimentos que pretende-se utilizar para testar a heurística, bem como para dar ao problema uma massa de dados para testes em futuras extensões desta pesquisa. Com esta cria-se um conjunto de instâncias com diferentes características, refletindo problemas de indústrias distintas e momentos distintos de mercado. Além da criação dos parâmetros de teste, é descrito o procedimento experimental, que consiste da descrição da forma como os métodos foram rodados, tanto pelos métodos propostos pelo autor, quanto pelo *software* de mercado (CPLEX). E, por fim, os resultados obtidos são mostrados.

No capítulo 6 é feita uma discussão dos resultados obtidos, compilando as tabelas de resultados mostrados no capítulo 5. São exploradas, principalmente, (i) as questões de complexidade matemática do problema formulado; e (ii) as diferenças de resultados obtidos perante os diversos fatores experimentais.

Finalmente, no capítulo 7 são apresentadas as conclusões deste trabalho, resumindo os principais pontos de discussão e retomando o problema de pesquisa proposto de forma a sumarizar as contribuições encontradas à literatura.

2. REVISÃO BIBLIOGRÁFICA

O objetivo deste terceiro capítulo é fazer uma revisão bibliográfica nos temas a que este trabalho se refere. Como ressaltado diversas vezes até aqui, está sendo explorada uma junção de dois temas clássicos da literatura, de forma que este capítulo se organiza três partes: inicialmente (i) serão abordados os avanços na área de corte e empacotamento, com ênfase maior no subproblema que diz respeito a este trabalho (*bin packing* bidimensional retangular); posteriormente, (ii) é feita uma revisão do problema de programação (*scheduling*), também dando uma atenção particular para o caso aqui tratado (problema em ambiente de máquina única, visando minimizar a soma das penalidades por atraso e adiantamento ponderadas, ou seja, $1||\sum_j \alpha_j E_j + \beta_j T_j$); e, por fim, (iii) são mencionados os trabalhos na linha da unificação entre os dois temas, seja por unir os objetivos de atendimentos de datas e aproveitamento do material, seja por considerar outros aspectos da programação no problema de corte.

2.1. Corte e empacotamento

Situações de formação de lotes, em geral, recaem em uma linha da pesquisa operacional que têm ganhado grande importância nas últimas décadas, os chamados problemas de corte e empacotamento (C&E). Uma evidência clara são as várias, e cada vez mais frequentes, edições especiais neste tema do periódico *European Journal of Operational Research*, de grande relevância e impacto na área (OLIVEIRA e WÄSCHER, 2007; WANG e WÄSCHER, 2002; BISCHOFF e WÄSCHER, 1995; DYCKHOFF e WÄSCHER, 1990).

De acordo com Bischoff e Wäscher (1995), o interesse crescente na pesquisa de corte e empacotamento como um todo pode ser explicado (i) por sua grande aplicabilidade; (ii) pela diversidade de problemas reais; e (iii) por sua grande complexidade.

Sua vastidão de aplicações é, de fato, facilmente verificável, tanto em aplicações imediatas (indústria de aço, vidro, papel, alumínio etc.), como também em situações semelhantes que recaem na mesma formulação teórica. Conforme já comentado, alguns exemplos deste último caso são balanceamento de linhas, formulação de orçamentos e *layout* de unidades de produção (BISCHOFF e WÄSCHER, 1995), arranjo de publicações de imprensa (DOWNSLAND e DOWNSLAND, 1995) e no planejamento logístico, que herda

características do problema visando a minimização de espaços ociosos na armazenagem, movimentação e transporte (GOLDEN, 1976 apud MIURA, 2008).

A estrutura básica de problemas de corte e empacotamento, segundo Wäscher, Haußner e Schumann (2007), compreende um conjunto de pequenos itens demandados (*outputs*) que devem ser agrupados em subgrupos (padrões de corte) a serem alocados em um conjunto de grandes objetos (*inputs*), obedecendo as restrições geométricas de não sobreposição e posicionamento completo nestes últimos, de forma a otimizar uma função-objetivo.

A ramificação da questão de formação de lotes em subclasses também fica em clara evidência ao analisar a literatura: problemas de diferentes dimensionalidades, homogeneidade de itens, tipos de unidades de corte e empacotamento, restrições de corte entre outras peculiaridades combinadas fazem com que o campo de origem a inúmeras abordagens distintas. Artigos que propõe tipologias de problemas de corte e empacotamento, como Wäscher, Haußner e Schumann (2007) e Dyckhoff (1990) mostram não só esta diversidade, conforme análise detalhada em item posterior, como também o volume de publicações em cada um destes subproblemas.

Em que se pese a enunciação física bastante simples do problema, sua complexidade é, com efeito, alta suficiente para classificá-lo como NP-difícil em termos de processamento, mesmo para os casos simplificados com unidimensionalidade, conforme Dyckhoff (1990), ou ainda para o caso de itens retangulares, conforme Downslan e Downslan (1992).

2.1.1. Tipologia

Uma primeira tipologia para este tipo de problema é a proposta por Dyckhoff (1990), que apresenta quatro critérios para definir um problema de corte e empacotamento:

- Dimensionalidade: 1 dimensão (1); 2 dimensões (2); 3 dimensões (3); ou n dimensões (N);
- Tipo de atribuição: todos os grandes objetos e uma seleção de pequenos itens (B); ou todos os pequenos itens e uma seleção de pequenos objetos (V);
- Conjunto de grandes objetos: um objeto (O); vários objetos idênticos (I); vários objetos distintos (D);

- Conjunto de pequenos itens: poucos itens de diferentes formatos (F); muitos itens de muitos formatos (M); muitos itens de relativamente poucos formatos (R); ou itens iguais/congruentes (C).

Embora reconheçam a importância desta tipologia para a literatura de corte e empacotamento, Wäscher, Haubner e Schumann (2007) apontam falhas na proposta: (i) restam lacunas de classificação de problemas clássicos (como o carregamento de veículos, *Vehicle Loading Problem*); (ii) há inconsistências entre a proposta e classificações usuais na literatura (como o caso bidimensional em que uma das dimensões do grande objeto é infinita, ou problema de corte de bobinas); (iii) as classes encontradas não são necessariamente homogêneas (especialmente no que diz respeito ao terceiro critério); (iv) além de uma dificuldade de entendimento linguístico decorrente de abreviações não provenientes do inglês.

Wäscher, Haubner e Schumann (2007) apresentam, portanto, uma tipologia melhorada para este tipo de problemas, inserindo novos critérios que abarcassem as novas abordagens surgidas nos últimos anos. Os novos critérios são:

- Dimensionalidade: 1 dimensão (1); 2 dimensões (2); 3 dimensões (3). Problemas n-dimensionais são vistos como variantes;
- Tipo de atribuição: maximização de outputs (valor dos itens alocados); ou minimização de inputs (valor dos objetos utilizados);
- Conjunto de pequenos itens: idênticos; fracamente heterogêneos; e fortemente heterogêneos;
- Conjunto de grandes objetos: um objeto, podendo ter dimensões fixas ou uma ou mais dimensões variáveis; e vários objetos, podendo ser idênticos, fracamente heterogêneos ou fortemente heterogêneos;
- Formato dos pequenos itens: regulares (podendo ser retângulos, círculos, caixas, cilindros, bolas, etc.); ou irregulares (não-regulares).

Partindo desta classificação, são delineados grupos básicos de problemas para cada dimensão, determinados pelo tipo de atribuição e conjunto de pequenos itens. A Figura 2.1 mostra essa hierarquização.

Figura 2.1 – Hierarquização dos problemas de corte e empacotamento (Fonte: Wäscher, Haubner e Schumann (2007))

Uma sexta característica não abordada em Wäscher, Haubner e Schumann (2007), porém muito encontrada em problemas desta natureza é a restrição de corte, ou seja, se existe trata-se de um corte guilhotinado ou não.

Arenales e Morábito (1995) definem um corte guilhotinado como aquele que produz dois novos retângulos e um padrão de corte guilhotinado como aquele que obtido pela sucessão de cortes guilhotinados. A Figura 2.2 mostra um corte guilhotinado e um padrão de corte guilhotinado.

Figura 2.2 – Exemplos de padrões de corte com restrições de guilhotinagem (Fonte: Arenales e Morábito, 1995)

Também é definido um corte não-guilhotinado de primeira ordem como aquele que produz cinco novos retângulos de forma que não formem um padrão de corte guilhotinado. A Figura 2.3 mostra exemplos de padrões de corte não-guilhotinados de primeira ordem.

Figura 2.3 – Exemplos de padrões de corte não-guilhotinados de primeira ordem (Fonte: Arenales e Morábito, 1995)

Importante ressaltar, embora seja trivial, que quando não há restrição de guilhotinagem, pode-se formar padrões das duas natureza. Em oposição, quando existe restrição de guilhotinagem, apenas pode-se formar padrões guilhotinados. Encontra-se na literatura outras variantes de restrições, como, por exemplo, o número de cortes guilhotinados permitidos na formação de um padrão (chamados de estágios).

Outra pequena variação comum nos problemas de corte e empacotamento é o comumente chamado de problema restrito (*constrained*). Neste tipo de problema não se admite produção de itens acima das quantidades demandadas. São, em geral, considerados quando o foco da pesquisa é a escolha de padrões de corte pré-definidos ou gerados, pois neste caso pode-se produzir mais itens de um determinado tipo do que o necessário.

Finalmente, existe mais uma característica para o problema bidimensional que contempla a possibilidade de rotação dos itens no objeto. Lodi, Martello e Vigo (1999) detalham que existem problemas em que os itens são orientados (O), ou seja, rotações não são permitidas; e outros em que é permitida a rotação destes (R). A restrição de rotação reflete, por exemplo, a necessidade de garantir uma determinada direção de grão do material cortado, para preservar suas propriedades físicas.

2.1.2. *Bin packing* bidimensional orientado com corte livre (2DBPIOF)

Como se trata do subproblema que concerne a este trabalho, aqui será dada uma importância maior a esta subcategoria, tendo uma revisão bibliográfica mais extensa e detalhada em relação a formulações, métodos exatos, *lower bounds* e heurísticas.

2.1.2.1. Descrição do problema

No problema de *bin packing* bidimensional (2BP ou 2DBP), segundo Lodi, Martello e Vigo (2002), são dados (i) um conjunto de itens retangulares $i \in I$, cada um tendo um comprimento (l_i) e uma largura (w_i); e (ii) um número ilimitado de objetos (*bins*) idênticos e finitos, com comprimento (L) e largura (W) determinados. Com estas entidades, o problema consiste na alocação, sem sobreposição de todos os itens no menor número de *bins* de maneira ortogonal. Na tipologia de Wäscher, Haubner e Schumann (2007), este problema encaixa-se na classificação de *Single Bin Size Bin Packing Problem* (SBSBPP).

Lodi, Martello e Vigo (1999) propõem uma tipologia simplificada deste classe de problemas em 4 subdivisões de acordo com a liberdade para rotação e para formato do corte, que já foram descritos na subseção 3.1.1. Assim, tem-se a seguinte notação:

- 2DBPIOG: itens têm orientação fixa e o corte é guilhotinado;
- 2DBPIRG: itens podem ser rotacionados em 90° e o corte é guilhotinado;
- 2DBPIOF: itens têm orientação fixa e o corte é livre (não-guilhotinado ou guilhotinado);
- 2DBPIRF: itens podem ser rotacionados em 90° e o corte é livre (não-guilhotinado ou guilhotinado).

Como já mencionado, o foco deste trabalho e, portanto, desta revisão é a classe de problemas de *bin packing* bidimensional orientado com corte livre (2DBPIOF).

2.1.2.2. Formulação matemática

Existem diversas abordagens para modelar através de programação linear inteira ou inteira mista o problema de *bin packing*. Aqui, serão mostradas três delas:

- i. A primeira discretiza as entidades do problema (itens e objetos) em unidades elementares (em geral, quando as dimensões são números inteiros, coincide com a unidade de medida) e é tratado como um problema de alocação destes pontos de forma contígua coerentemente com as dimensões das entidades. Haddjiconstantinou e Christofides (1995) abordam o problema desta maneira, porém para o problema de maximizar o valor em 1 *bin*;
- ii. Outra possibilidade é o tratamento contínuo das variáveis dimensionais utilizando variáveis binárias para criar disjunções de não-sobreposição entre itens. Sawaya e Grossmann (2005) utilizam tal abordagem para o problema de *strip packing* bidimensional, enquanto Hifi et al. (2010) para o problema de *bin packing* tridimensional. Será apresentado o modelo de Hifi et al. (2010), que é facilmente adaptável para seu análogo em 2 dimensões;
- iii. Finalmente, a formulação por geração de colunas é bastante consagrada para o problema de *bin packing*, desde suas primeiras versões na década de 60 em Gilmore e Gomory (1965), que considera o problema de corte multiestágio guilhotinado. Morábito (1994) revisita este modelo em uma aplicação para a indústria moveleira. O método da geração de colunas é baseado na otimização da escolha de padrões de corte pré-definidos. Como o número de padrões de corte é potencialmente muito grande, o método inicia com alguns padrões elementares e é enriquecido a cada resolução com novos padrões de corte (colunas).

O Quadro 2.1 mostra a formulação discretizada de Beasley (1985), considerando vários tipos de objetos em estoque. Seja n o número de objetos diferentes i , com comprimento L_i ($i = 1, \dots, n$), largura W_i ($i = 1, \dots, n$), custo associado ao seu uso unitário de f_i ($i = 1, \dots, n$), custo por unidade de área não utilizada em nenhum item de c_w e um número máximo de objetos diferentes a serem utilizados k . Seja m o número de itens diferentes a serem empacotados, com comprimento l_j ($j = 1, \dots, m$), largura w_j ($j = 1, \dots, m$), valor associado v_j ($j = 1, \dots, m$), limitante inferior de retângulos j cortados a_j ($j = 1, \dots, m$), limite superior de

tetângulos j cortados b_j ($j = 1, \dots, m$). Seja $P(i)$ um conjunto de padrões de corte bidimensionais para um objeto do tipo i ; e d_{jpi} o número de peças j que ocorrem em um padrão p pertencente a $P(i)$. São variáveis de decisão para o problema x_{pi} , o número de vezes que o padrão p pertencente a $P(i)$ é usado para cortar um objeto de estoque do tipo i ; u_j , o número de peças retangulares j que são cortadas para atender à demanda deste tipo de peça; y_i , um binário que vale 1 se um objeto do tipo i é usado e 0 caso contrário.

$\min$	$\sum_{i=1}^n \sum_{p \in P(i)} (f_i + c_w L_i W_i) x_{pi} - \sum_{j=1}^m (v_j - c_w l_j w_j) u_j$	(2.1)
$s. a.$	$a_j \leq u_j \leq b_j$	$j = 1, \dots, m$ (2.2)
	$u_j \leq \sum_{i=1}^n \sum_{p \in P(i)} d_{jpi} x_{pi}$	$j = 1, \dots, m$ (2.3)
	$\sum_{i=1}^n y_i \leq k$	(2.4)
	$\sum_{p \in P(i)} x_{pi} \leq M y_i$	$i = 1, \dots, n$ (2.5)
	$y_i \in \{0,1\}$	$i = 1, \dots, n$ (2.6)
	$x_{pi} \in \mathbb{Z}^+$	$i = 1, \dots, n, \forall p \in P(i)$ (2.7)
	$u_j \in \mathbb{Z}^+$	$j = 1, \dots, m$ (2.8)

Quadro 2.1 – Formulação discretizada do 2DBPP (Beasley, 1985)

A função objetivo (2.1) visa minimizar o custo da solução gerada. O primeiro termo da expressão representa o custo fixo dos objetos em estoque usados com suas penalidades por área, enquanto o segundo termo reduz essa penalidade com o valor dos itens utilizados. Em (2.2) é assegurado o atendimento da demanda dos itens. Em (2.3), o número de peças do tipo j produzido deve estar restrito superiormente pela quantidade de itens deste que estão constando nos padrões de corte utilizados. A inequação (2.4) assegura que um máximo de k objetos são utilizados, enquanto (2.5) impõe que um padrão de corte só é gerado com um certo objeto em estoque se este é efetivamente utilizado, dando valor físico à variável y_i (aqui

M é um número suficientemente grande). As relações (2.6), (2.7) e (2.8) mostram o domínio das variáveis.

No segundo grupo de formulações, são utilizadas variáveis contínuas para as dimensões das entidades e variáveis binárias representando disjunções para não-sobreposição. Chen et al. (1995) propuseram uma formulação com esta característica, utilizada por diversos outros autores como Hifi et al. (2010), esta última mostrada no Quadro 2.2. Embora este modelo seja tridimensional, é facilmente adaptável para o caso bidimensional. Isso é feito mais adiante, no capítulo 4, pois tal modelo servirá de base para a extensão com considerações de *scheduling*.

Seja um conjunto de n itens i ($i = 1, \dots, n$) com dimensões (w_i, h_i, d_i) e *bins* de dimensões (W, H, D) com rótulos inteiros γ_i . As variáveis de decisão x_i, y_i e z_i são as coordenadas do canto inferior direito e ao fundo do item i . As variáveis binárias l_{ij} valem 1 se o item i está totalmente à esquerda do item j e 0, caso contrário; u_{ij} e b_{ij} são análogas para os eixos y e z . A variável binária c_{ij} vale 1 se o *bin* do item i tem rótulo precedendo ao rótulo do *bin* em que está o item j . O maior rótulo de *bin* utilizado (que coincide com o número de *bins* diminuído de 1 unidade) é γ , enquanto $\bar{\gamma}$ é um *upper bound* do número de *bins* n (2.9) s.

$(MILP1) \quad \min \quad \gamma$	
$s.c. \quad l_{ij} + l_{ji} + u_{ij} + u_{ji} + b_{ij} + b_{ji} + c_{ij} + c_{ji} = 1,$	$i < j = 1, \dots, n, \quad (2.9)$
$x_i - x_j + W(l_{ij} - c_{ij} - c_{ji}) \leq W - w_i,$	$i \neq j = 1, \dots, n, \quad (2.10)$
$y_i - y_j + H(u_{ij} - c_{ij} - c_{ji}) \leq H - h_i,$	$i \neq j = 1, \dots, n, \quad (2.11)$
$z_i - z_j + D(b_{ij} - c_{ij} - c_{ji}) \leq D - d_i,$	$i \neq j = 1, \dots, n, \quad (2.12)$
$(\bar{\gamma} - 1)(l_{ij} + l_{ji} + u_{ij} + u_{ji} + b_{ij} + b_{ji})$	
$+ \gamma_i - \gamma_j + \bar{\gamma}c_{ij} \leq \bar{\gamma} - 1,$	$i \neq j = 1, \dots, n, \quad (2.13)$
$l_{ij}, u_{ij}, b_{ij}, c_{ij} \in \{0, 1\},$	$i \neq j = 1, \dots, n,$
$0 \leq x_i \leq W - w_i,$	$i = 1, \dots, n,$
$0 \leq y_i \leq H - h_i,$	$i = 1, \dots, n,$
$0 \leq z_i \leq D - d_i,$	$i = 1, \dots, n,$
$0 \leq \gamma_i \leq \gamma \leq \bar{\gamma},$	$i = 1, \dots, n.$

Quadro 2.2 – Formulação contínua com restrições disjuntivas para o 3DBPP (Hifi et al., 2010)

A função objetivo visa minimizar o maior rótulo de um objeto (*bin*), o que traduz uma minimização do número de *bins* utilizados. A restrição (2.9) garante que dois itens não

estejam sobrepostos no posicionamento de nenhuma de suas 3 dimensões ou que estejam em objetos diferentes ($c_{ij} = c_{ji} = 0$). As restrições (2.10) a (2.12) retratam fisicamente as restrições de não-sobreposição, para que estejam refletidas no significado das variáveis l_{ij} , u_{ij} e b_{ij} . Finalmente, a restrição (2.13) garante que $c_{ij} = c_{ji} = 0$ quando dois itens estão no mesmo *bin*, enquanto $c_{ij} = 1$ se o item i estiver posicionado em um *bin* anterior a j , e vice-versa. As demais restrições são restrições de binaridade e garantem que os itens sejam posicionados nos limites dos objetos.

Finalmente, no Quadro 2.3 temos o modelo apresentado por Morábito (1994) para o mesmo problema, porém com a abordagem de escolha de uma certa quantidade de padrões de corte, utilizando como método a geração de colunas.

Seja D um conjunto denominado demanda por cada um dos itens i , sendo b_i a quantidade demandada. O parâmetro x_{ip} denota a quantidade de peças do item i no padrão de corte p , de forma que cada vetor (coluna) constituído pelos x_{ip} com um mesmo p formam um padrão de corte. A variável de decisão y_p é inteira e denota a quantidade de padrões de corte do tipo p que serão escolhidos.

$\min \quad Z = \sum_{p=1}^P y_p \quad (2.14)$
$s. a. \quad \sum_{p=1}^P x_{ip} y_p = b_i \quad i \in D, \quad (2.15)$
$y_p \in \mathbb{Z}^+ \quad p = 1, \dots, P \quad (2.16)$

Quadro 2.3 – Formulação para escolha de padrões de corte para resolução por geração de colunas (Morábito, 1994)

A função objetivo (2.14) é minimizar o número de padrões de corte utilizados (ou seja, o número de *bins*). A restrição (2.15) visa o atendimento da demanda através dos padrões de corte escolhidos, enquanto (2.16) garante que o número de padrões de corte é inteiro.

Apesar da simplicidade desta formulação, a complexidade reside justamente na geração das colunas (padrões de corte) para o problema. Como a quantidade de possíveis padrões de corte é imensa para um pequeno conjunto de itens, é inviável esgotar a totalidade de alternativas

por enumeração explícita. Ao invés disto, são utilizados métodos (*branch-and-price*) para gerar novas colunas que têm potencial de aumentar o valor da solução.

2.1.2.3. *Lower bounds*

Como visto anteriormente, qualquer das formulações do 2DBP|OIF (ou mesmo as outras variantes do 2DBP) apresenta um espaço combinatório grande. De fato, Lodi, Martello e Vigo (2002) afirmam que mesmo o 1DBP é um problema fortemente NP-difícil, de forma que o 2DBP mantém tal condição.

Desta forma, métodos de programação linear e algoritmos exatos restringem-se a um determinado porte de problemas, que nem sempre alcançam a quantidade de itens almejada por instâncias reais.

Desta forma, é útil desenvolver, através das propriedades conhecidas do problema, um limitante inferior para a solução ótima (*lower bound*). Quanto mais eficiente este valor (ou seja, quanto mais alto, já que se trata de um problema de minimização), melhor a avaliação de soluções heurísticas (*upper bounds*) ou, ainda, mais eficientemente utiliza-se um método exato.

A primeira noção de *lower bound* para o 2DBP é o limitante contínuo, ou seja, o arredondamento para cima da razão entre a área total dos itens e a área de um *bin* (como pode ser visto a seguir, sendo L_0 o valor do limitante inferior, L o comprimento do *bin*, W sua largura e l_i e w_i as dimensões de cada um dos itens i). É intuitivo que a solução ótima não pode ser inferior a este valor.

$$L_0 = \left\lceil \frac{\sum_i l_i w_i}{LW} \right\rceil$$

Martello e Vigo (1998) mostram que o pior caso de performance deste limitante é $\frac{1}{4}$, o que seja, seja uma instância D e $OPT(D)$ o seu ótimo, tem-se que:

$$\frac{L_0(D)}{OPT(D)} \geq \frac{1}{4}$$

Serairi e Haouari (2010) fazem uma comparação computacional de métodos para obtenção de limitantes inferiores para o problema de *bin packing* bidimensional, dividindo os objetos de estudo em 3 famílias:

- A primeira família aplica métodos de modificação dos tamanhos dos itens considerando apenas uma das dimensões. Estão nesta família dois dos métodos descritos em Martello e Vigo (1998 apud Serairi e Haouari, 2010) e, ainda, um dos métodos descritos em Boschetti e Mingozzi (2002), sendo que este último domina os demais, porém com complexidade maior. É proposto um processo chamado de redução da dimensão do item, em que este parâmetro é atualizado (aumentado) sem prejudicar a solução ótima para o problema, porém melhorando a obtenção de limitantes inferiores;
- A segunda família já modifica ambas dimensões de cada item, seguindo o mesmo princípio, de forma a aumentar suas dimensões sem criar um sub-ótimo para o problema de *bin packing*. Posteriormente, são computados limitantes para problemas de *bin packing* unidimensional. São classificados como estando nesta família outro método de Martello e Vigo (1998), três dos métodos propostos em Boschetti e Mingozzi (2002) e um dos métodos propostos por Carlier, Clautiaux e Moukrim (2007);
- Finalmente, a terceira família, na qual se encaixam os métodos de Fekete e Shepers (2000b) e outro método de Carlier, Clautiaux e Moukrim (2007), constitui-se daqueles que são baseados em funções viáveis duais (*dual feasible functions* ou DFF).

Um exemplo de processo de modificação de itens por redução de dimensão é explicado por Boschetti e Mingozzi (2002). Seja um item a_j com largura w_j , pertencendo a um conjunto de itens J a ser empacotado em um *bin* de largura W . Esta largura pode ser atualizada, sem perda de generalidade para $w_j + (W - W_j^*)$, sendo que:

$$W_j^* = \max \{ w = \sum_{k \in J \setminus \{j\}} w_k \xi_k + w_j : w \leq W, \xi_k \in \{0,1\}, k \in J \setminus \{j\} \}$$

Em outras palavras, esta relação diz que se a maior combinação de larguras do conjunto com o item em questão não atinge a largura do *bin*, a diferença entre a largura do *bin* e esta combinação máxima pode ser acrescida à largura do item sem perda de generalidade. Este

processo em específico está melhor ilustrado e explicado em Boschetti, Mingozzi e Hadjiconstantinou (2002).

A utilidade deste processo de redução fica clara, por exemplo, em um dos *lower bound* proposto por Martello e Vigo (1998 apud Serairi e Haouari, 2010) para o problema de *bin packing* unidimensional, em que os itens são divididos em grandes, médios e pequenos e é possível fazer inferências melhores que o limitante contínuo de quantos *bins* serão necessários pelo fato de que itens grandes ocupam um *bin* para si próprios necessariamente.

Fekete e Schepers (2000b) trabalharam com limitantes para dimensionalidades superiores a 1, introduzindo o uso das funções viáveis duais (DFF). Uma DFF é uma função $u: [0, 1] \rightarrow [0, 1]$ em que, para um conjunto S de números reais não negativos, tem-se a seguinte relação:

$$\sum_{x \in S} x \leq 1 \Rightarrow \sum_{x \in S} u(x) \leq 1$$

Fekete e Schepers (1998) mostram que se A é um conjunto de itens com dimensões $(x_1, \dots, x_n)$, então um limitante inferior para a instância $u(a)$ é um limitante inferior para a instância A . Isto implica que funções adequadas podem ser exploradas para melhorar limitantes de certas instâncias.

Finalmente, Carrier, Clautiaux e Moukrim (2007) – cujo método é apontado no estudo de Serairi e Haouari (2010) como o consistentemente mais eficiente da literatura até então – trabalham com o conceito de funções identicamente viáveis (*identically-feasible functions*), que, ao mesmo tempo mantém o conceito de aumentar itens grandes como os anteriores, também procura remover itens cujas dimensões são tão pequenas que não influenciam no ótimo. O conceito de uma função identicamente viável é justamente aquela que, se aplicada a todos os itens de uma instância, não altera seu ótimo.

Um exemplo de função identicamente viável mostrada pelos autores é a seguinte: seja D uma instância para o problema de *bin packing* bidimensional e p um número inteiro maior ou igual a 1 e menor ou igual a metade da altura do *bin* (H). Para cada item i com altura maior do que H subtraído de p ($H - p$) é criado um novo *bin* com a largura deste item e a altura residual entre sua altura e a altura do bin original (ou seja, largura w_i e altura $H - h_i$). Se existe como

empacotar os itens chamado de pequenos (altura menor do que p) nestes *bins* criados, então uma função identicamente viável é:

$$v_1^p: A \rightarrow A^1,$$

$$a_i \mapsto a'_i \quad \text{onde} \quad \left\{ \begin{array}{l} w'_i = w_i \text{ e } h'_i = H \text{ se } h_i > H - p \\ w'_i = 0 \text{ e } h'_i = 0 \text{ se } h_i < p \\ w'_i = w_i \text{ e } h'_i = h_i \text{ caso contrário} \end{array} \right\}$$

Em outras palavras, se a condição anterior existe, os itens chamados de pequenos podem ser considerados inexistentes e os grandes tomar a altura do *bin*. Os demais permanecem como estão. Essa função está ilustrada na Figura 2.4.

Figura 2.4 – Ilustração da função identicamente viável exemplificada
(Fonte: Carlier, Clautiaux e Moukrim, 2007)

2.1.2.4. Métodos exatos

Métodos exatos são aqueles que obtêm a solução ótima e garantem sua otimalidade. Para problemas NP-completos (como o que está sendo estudado), algoritmos exatos têm tempo de execução não-polinomial (TALBI, 2009).

Martello e Vigo (1998 apud Serairi e Haouari, 2010) utilizam os *lower bounds* propostos em seu artigo, até então inéditos na literatura, para criar um método de *branch-and-bound* que pode resolver instâncias de porte pequeno.

Fekete e Schepers (2000a) propõem uma representação de uma solução para um problema de *bin packing* bidimensional através de grafos de suas projeções em que os itens são nós e os arcos são sobreposições naquela projeção. A Figura 2.5 ilustra tal representação. Sua vantagem é que empacotamentos equivalentes são agrupados em classes de empacotamentos, de forma que métodos de exploração enumerativos tomam uma dimensão menos exaustiva com tal representação. Assim, em Fekete e Schepers (2000c), esta notação é utilizada para a construção de um procedimento *branch-and-bound* em que esta notação é utilizada para a procura nos ramos; enquanto em Fekete, Schepers e van der Veen (2007) uma nova busca em árvore é proposta.

Figura 2.5 – Ilustração da representação de uma solução para o problema de *bin packing* bidimensional por teoria dos grafos (Fonte: Fekete e Schepers, 2000a)

Recentemente, avanços foram feitos em problemas correlatos por Clautiaux, Carlier e Moukrim (2007b), que oferecem métodos exatos para o problema de verificar se um conjunto de itens pode ser empacotado em um único *bin*, melhorando o procedimento de *branch-and-bound* proposto por Martello e Vigo (1998 apud Serairi e Haouari, 2010) e, posteriormente, sugerindo uma relaxação para o problema. Já Dolatabadi, Lodi e Monaci (2012) trabalham com o problema de maximização do valor dos itens empacotados em um *bin* com corte guilhotinado (problema da mochila bidimensional guilhotinada), utilizando um algoritmo recursivo em que dado um conjunto de itens em um *bin*, os cortes guilhotinados associados são construídos.

2.1.2.5. Heurísticas construtivas

Heurísticas construtivas são aquelas em que se inicia com uma solução vazia, sendo construída por uma sucessão de atribuições de valores a uma variável de decisão de cada vez, até que uma solução completa é gerada (TALBI, 2009).

Para o problema de *bin packing*, Lodi, Martello e Vigo (2002) revisam este tipo de método de solução, classificando-as em três grupos:

- Algoritmos em uma fase: posicionam os itens nos bins de acordo com uma regra gulosa, como por exemplo o *Finite Next-Fit* (FNF) ou *Finite First-Fit* (FFF), em que os itens são ordenados decrescentemente em altura e posicionados respectivamente no próximo ou no primeiro *bin* em que couberem em um certo nível;
- Algoritmos em duas fases: utilizam uma regra parecida com a anterior, porém existe uma primeira fase de alocação em nível e uma fase posterior de alocação destes níveis nos *bins*. São exemplos, segundo a regra de posicionamento o *Hybrid First-Fit* (HFF), o *Hybrid Best-Fit* (HBF, em que o item é posicionado no nível em que a sobra até a largura do *bin* é mínima), o *Floor-Ceiling* (FC, em que o processo de alocação dos itens nos níveis é feito tanto no canto inferior esquerdo quanto no superior direito do nível). A Figura 2.6 ilustra o processo de alocação nos níveis para o HFF e o HBF, a Figura 2.7 ilustra o processo de alocação de níveis em *bins* (segunda etapa), enquanto a Figura 2.8 mostra a primeira etapa no *Floor Ceiling*;

Figura 2.6 – Ilustração dos processos de alocação, *First-Fit Decreasing Height* e *Best-Fit Decreasing Height* (Fonte: Lodi, Martello e Vigo, 2002)

Figura 2.7 – Ilustração dos processos de alocação de níveis nos bins
(Fonte: Lodi, Martello e Vigo, 2002)

Figura 2.8 – Ilustração dos processos de alocação de itens no Floor Ceiling
(Fonte: Lodi, Martello e Vigo, 2002)

- Algoritmos não em nível: a lógica de posicionamento no bin não é feita em nível. Isso acontece, por exemplo, no *Bottom-Left* (BL, em que o item é posicionado alinhado à esquerda no canto mais inferior disponível) e no *Alternate Directions* (AD, em que um primeiro conjunto de itens é alocado seguindo a regra *Best-Fit Decreasing Height* e os demais de maneira alternada da esquerda para a direita ou o oposto).

A classe dos algoritmos em dois níveis mostra também que algoritmos para *strip packing* podem ser aproveitados de forma a constituir a primeira fase. Um exemplo com resultados muito interessantes é o BFDH* em Bortfeldt (2006), que utiliza a base do BFDH com a modificação de permitir que os espaços livres acima dos itens posicionados em um nível sejam preenchidos com o item de maior área não alocado.

Morabito e Pureza (2010) propõe uma abordagem baseada em programação dinâmica e grafos E-OU para o problema bidimensional guilhotinado de demanda restrita. Mais recentemente, Liu, Chu e Wang (2011) revisitam a ideia, propondo uma heurística flexível em que, para cada item posicionado, novos possíveis retângulos para posterior alocação são gerados, como mostra a Figura 2.9. Além de obter bons tempos computacionais em geral, uma grande quantidade de instâncias clássicas na literatura pode ser resolvida de maneira ótima.

Figura 2.9 – Ilustração do processo utilizado para o raciocínio de programação dinâmica (Fonte: Liu, Chu e Wang, 2011)

2.2. Programação da produção (*scheduling*)

Pinedo (2008) define *scheduling* como o processo de tomada de decisão, usado sistematicamente em manufatura e serviços, que lida com a alocação de recursos a tarefas em um dado período, almejando otimizar um ou mais objetivos. Já Wight (1984) enxergam o mesmo problema sob a lógica do estabelecimento do *timing* para desempenho de uma tarefa, ou seja, o que deve ser feito primeiro e com que recursos.

Henry Gantt foi pioneiro na área do sequenciamento, no início do século XX, embora publicações sobre o assunto tenham ocorrido somente ao final da década de 50. Trabalhos

como o de Kelley e Walker (1959) propondo o algoritmo de caminho crítico para redes de precedência (PINEDO, 1995) são destacados nos primórdios da literatura da área. Ovacik e Uzsoy (1997) atribuem a crescente importância do sequenciamento como um subproduto do foco na satisfação do cliente trazida por metodologias como o TQM (Total Quality Management). Este novo ambiente de produção forçou a necessidade de reduzir *lead times* e melhorar o desempenho de entregas, mesmo que internas. Pinedo (1995) ressalta, ainda, a importância de uma boa programação em empresas com datas compromissadas, pois seu não-atendimento resulta em perda de imagem.

Mesmo assim, Herrmann (2006) aponta que poucas empresas fabris tiram vantagens dos avanços na área de sequenciamento, já que este potencializou-se recentemente com o avanço da capacidade de processamento informático. Isso pode ser atribuído à complexidade em geral associada a tal tipo de problemas.

Em termos de modelagem, Pinedo (2008) caracteriza o problema com uma notação particular, em que tem-se um conjunto de m máquinas ($i = 1, \dots, m$) e um conjunto de n ordens ($j = 1, \dots, n$) que apresentam um roteiro de produção, ou seja, um conjunto de operações (i, j) . Cada operação é caracterizada por um tempo de processamento p_{ij} , sendo que existe uma expectativa do seu instante de término, que é o chamado prazo, data de entrega ou *due date* (d_j). Cada ordem também apresenta um instante de liberação (r_j), que é o momento em que está pronta para ser iniciada. Também é uma característica da ordem o seu peso ou importância no sistema (w_j).

2.2.1. Tipologia

Os problemas de *scheduling* apresentam uma descrição tríplice proposta inicialmente por Conway, Maxwell e Miller (1967), que foi refinada nas últimas décadas e tem-se um paradigma atual na proposta de Lawler, Lenstra e Rinnooy Kan (1982), com pequenas diferenças de notação.

Tal classificação baseia-se na tríplice $\alpha | \beta | \gamma$. Nela, o índice α caracteriza o ambiente de produção, o índice β denota detalhes, características e restrições particulares do problema e, por fim, o índice γ descreve o objetivo a ser otimizado.

2.2.1.1. Ambiente de produção (α)

O ambiente de produção (α) é uma entrada sempre única (um sistema só possui uma configuração, necessariamente). Pinedo (2008) sugere as seguintes possibilidades:

- Máquina única (1): ambiente mais simples, em que o processo tem um passo único e um único recurso em que pode ser alocado, como ilustrado na Figura 2.10;

Figura 2.10 - Ilustração do ambiente de máquina única ($\alpha = 1$) (Fonte: elaboração do autor)

- Máquinas idênticas em paralelo (P_m): novamente o processo apresenta passo único, porém apresenta recursos alternativos para sua execução, como ilustrado na Figura 2.11. Tais recursos são idênticos, ou seja, o tempo de processamento p_j de cada item é o mesmo em qualquer máquina;

Figura 2.11 - Ilustração do ambiente de máquinas em paralelo ($\alpha = P2$) (Fonte: elaboração do autor)

- Máquinas em paralelo com velocidades proporcionais (Q_m): trata-se de um caso mais genérico do que o anterior, em que as máquinas em paralelo apresentam velocidades de processamento distintas, porém mantém uma proporcionalidade entre si para cada item (ou seja, $p_{ij} = p_j/v_i$);

- Máquinas em paralelo com velocidades diferentes não relacionadas (R_m): caso mais genérico que o anterior, em que a velocidade da máquina depende do item em processamento, ou seja existe os tempos p_{ij} não guardam nenhuma relação entre si;
- Flow shop (F_m): ambiente em que existem m máquinas em série, como ilustrado na Figura 2.12, sendo que cada item deve ser processado em sequência em cada um destes m centros;

Figura 2.12 – Ilustração do ambiente *flow shop* ($\alpha = F3$) (Fonte: elaboração do autor)

- Flow shop flexível (FF_c): ambiente em que existem c estágios (centros de produção), havendo um certo número de máquinas em paralelo dentro de cada um destes estágios, como ilustrado na Figura 2.13;

Figura 2.13 - Ilustração do ambiente *flow shop flexível* ($\alpha = FF3$) (Fonte: elaboração do autor)

- Job Shop (J_m): neste ambiente, cada item apresenta um roteiro predeterminado em um conjunto de m máquinas, como ilustra a Figura 2.14;

Figura 2.14 - Ilustração do ambiente *job shop* ($\alpha = J6$) (Fonte: elaboração do autor)

- Job Shop flexível (FJc): analogamente ao *flow shop* flexível, trata-se de um caso genérico do *job shop* em que existem c centros, com um determinado número de máquinas idênticas em que é possível alocar o item que ali chega, conforme mostra a Figura 2.15;

Figura 2.15 – Ilustração do ambiente job shop flexível ($\alpha = \text{JF6}$) (Fonte: elaboração do autor)

- Open Shop (Om): finalmente, o ambiente de *open shop* é aquele em que a tarefa deve ser processada em um certo número de centros, porém não existe uma sequência de roteiro predeterminada, podendo tomar diferentes rumos conforme a conveniência da programação.

2.2.1.2. Características e restrições do sistema (β)

O segundo campo da classificação proposta é o de características e restrições (β), que pode apresentar múltiplas entradas. Existe uma grande quantidade de variações possíveis, dentre as quais, Pinedo (2008) enuncia:

- Datas de liberação ou release dates (r_i): considera que as datas de liberação são dados do problema (ou seja, as ordens não estão todas disponíveis no início);
- Preempção (prmp): possibilidade de interromper o processamento de uma operação antes de seu término para iniciar outra;
- Restrições de precedência (prec): necessidade de terminar uma tarefa antes de iniciar outra. Neste caso, distinguem-se três tipos de problema: chain (cada ordem precede um sucessor), intree (um conjunto de predecessor tem um único sucessor) ou outtree (um conjunto de sucessores tem um único predecessor);

- Setup dependente de sequência (s_{jk}): ocorrem quando o tempo de preparação de uma ordem depende do tempo de preparação da ordem que a precedeu na programação;
- Famílias de ordens (fmls): caso particular do anterior em que o tempo de preparação é dependente da família a qual o item pertence. Uma revisão desta condição pode ser vista em Potts e Kovalyov (2000);
- Processamento em lotes (batch(b)): ocorre quando a operação é feita simultaneamente em mais de uma ordem, sendo b o número de ordens que uma batelada comporta. Interessante observar que se aproxima do tema do trabalho, porém aqui não são consideradas restrições geométricas. Uma revisão desta condição pode ser vista em Potts e Kovalyov (2000);
- Breakdowns (brkdown): considera tempos de parada programados para as máquinas. É comum quando existe um programa de manutenção preventiva. Uma revisão desta condição pode ser vista em Lee (2004);
- Restrição de elegibilidade de máquina (M_j): nem todas as máquinas são capazes de processar todas as ordens;
- Permutação (prmu): em um *flow shop*, chama-se permutacional quando a sequência de operações em cada máquina é FIFO (First In, First Out);
- Bloqueio (block): trata-se da situação em que existe limitação de estoque intermediário entre as operações. É comum em células de manufatura, em que não existe espaço para tal armazenamento;
- No-wait (nwt): não é permitida a espera de uma ordem entre uma operação e a seguinte. Geralmente esta condição ocorre em máquinas que aproveitam temperaturas obtidas em operações anteriores ou quando há perecibilidade rápida do produto;
- Recirculação (rcrc): ocorre quando uma ordem pode visitar uma operação mais de uma vez em seu roteiro.

Há outras possibilidades, ainda, de outros campos, porém em geral são auto-explicativos. Por exemplo, $d_j = d$ implica que todas as datas de entrega (prazos) são iguais entre as ordens.

2.2.1.3. Objetivo (γ)

Finalmente, o terceiro campo diz respeito ao objetivo a ser perseguido (em geral, minimizado). Trata-se de um campo de entrada única, embora possa refletir múltiplos

objetivos. Também trata-se de um campo aberto a novas possibilidades, embora existam objetivos classicamente explorados na literatura.

Pacheco e Santoro (1999) pontuam como principais objetivos da programação em ambiente *job shop* o atendimento de prazos, minimização do tempo de fluxo ou estoques intermediários e maximização da utilização da capacidade disponível.

Dentre os objetivos mais comumente explorados, Pinedo (2008) aponta a minimização de:

- Horizonte de programação ou *makespan* ($C_{\max}$): é o instante de término da última ordem programada. Seu valor é relacionado com a boa utilização dos recursos;
- Desvio de datas ou *lateness*: ocorre tanto para atrasos (*tardiness*) quanto adiantamentos (*earliness*). Pode minimizar-se o maior desvio de datas, a soma dos desvios ou ainda o desvio ponderado. Alternativamente, pode-se objetivar a minimização da soma dos atrasos e adiantamentos ponderados (quando a penalidade para cada um é diferenciada) ou apenas um dos dois;
- Número de ordens atrasadas: como o próprio nome diz, trata-se da minimização da quantidade de ordens atrasadas, independentemente da intensidade dos atrasos;

2.2.2. Métodos de solução

Pacheco e Santoro (1999) hierarquizam as tratativas ao problema de programação em ambiente de *jobshop*, como pode ser visto na Figura 2.16.

Critérios de classificação			
1	Quanto à otimalidade	9	Quanto ao número e detalhe dos gargalos
2	Quanto à forma de enumerar as alternativas	10	Quanto à estratégia de descarte de ramos
3	Quanto ao número de passos	11	Quanto à possibilidade de diversificação
4	Quanto à forma de limitação de alternativas	12	Quanto à origem da sequência
5	Quanto à existência de instificação	13	Quanto à estatégia de intensificação e diversificação
6	Quanto ao foco do procedimento	14	Quanto à forma de obtenção de sequências
7	Quanto à forma de consideração do tempo	15	Quanto à inserção ou não de ociosidade
8	Quanto à estrutura de dados usada para busca		

Figura 2.16 – Classificação hierarquizada dos modelos de solução para o problema de *job shop scheduling* (Fonte: Pachecho e Santoro, 1999)

Tal hierarquia divide os métodos de solução entre otimizantes (atingem o ótimo) e heurísticos (buscam uma boa solução sem garantir o ótimo). Como boa parte dos problemas de *scheduling* são NP-completos ou NP-difíceis (ainda não existe uma forma de resolvê-los em tempo polinomial), poucos são os casos em que encontra-se um algoritmo otimizador eficiente. Os métodos chamados de otimização implícita são utilizados, porém chegam ao ótimo apenas em instâncias muito pequenas.

Os métodos heurísticos são primariamente divididos em métodos de busca e construtivos. A diferença primordial entre eles é que os **métodos de busca** partem de uma solução inicial e buscam melhorá-la. Isto pode ser feito com uma busca aleatória, uma busca em árvore, uma busca em vizinhança (heurísticas de melhoria) ou busca estendida (metaheurísticas). Já os **métodos construtivos** não partem de uma solução, mas sim a constroem. Estes são divididos em métodos orientados à sequência e orientados ao gargalo.

2.2.3. Programação em máquina única com objetivo de minimizar soma das penalidades por atraso e adiantamento ($1\|\sum_j \alpha_j E_j + \beta_j T_j$)

Dentro da tipologia apresentada, o subproblema de *scheduling* que será combinado neste trabalho terá uma maior importância, assim como feito anteriormente com o *bin packing bidimensional*. Será apresentada nas próximas subseções uma revisão bibliográfica mais extensa e detalhada em relação a formulações, métodos exatos, *lower bounds* e heurísticas deste subproblema.

2.2.3.1. Formulação matemática para máquina única ($\alpha = 1$)

Segundo Pinedo (2008), as restrições precedência temporal para o problema de scheduling podem ser modelados de três maneiras:

- Variáveis indexadas pelo tempo (*Time-indexed variables*): seja uma ordem j ($j = 1, \dots, n$) e um período t ($t = 0, \dots, T$), a variável x_{jt} é uma binária em que $x_{jt} = 1$ se a ordem j está alocada para iniciar no instante t e 0, caso contrário. As restrições que modelam as relações de precedência são mostradas em (2.17)-(2.19). Em (2.17) garante-se que cada ordem só tenha um início programado. Em (2.18) garante-se que em um dado instante de tempo, apenas uma ordem esteja alocada para processamento. A restrição (2.19) impões que a variável x_{jt} seja binária;

$$\sum_{t=0}^T x_{jt} = 1 \quad j = 1, \dots, n, \quad (2.17)$$

$$\sum_{j=1}^n \sum_{s=\max(t-p_j, 0)}^{t-1} x_{js} = 1 \quad t = 0, \dots, T, \quad (2.18)$$

$$x_{jt} \in \{0,1\} \quad j = 1, \dots, n, \quad t = 0, \dots, T \quad (2.19)$$

- Variáveis sequenciadoras (*Sequencing variables*): seja duas ordens j e k ($j, k = 1, \dots, n$), a variável x_{jk} é uma binária em que $x_{jk} = 1$ se a ordem j está alocada temporalmente antes de k e 0, caso contrário. As restrições que modelam as relações de precedência são mostradas em (2.20)-(2.23). Em (2.20) garante-se que se uma ordem j precede k , k não precede j , ou vice-versa. O conjunto de restrições (2.21) garante consistência entre precedências três-a-três (transitividade), ou seja, se uma ordem k precede j e outra ordem l precede k , então l tem que preceder j também. Em (2.21) garante-se que em um dado instante de tempo, apenas uma ordem esteja alocada para processamento. A restrição (2.22) impõe que a variável x_{jj} seja nula, enquanto (2.23) impões que a variável x_{jk} seja binária;

$$x_{kj} + x_{jk} = 1 \quad j, k = 1, \dots, n, \quad j \neq k \quad (2.20)$$

$$x_{kj} + x_{lk} + x_{jl} \geq 1 \quad j, k, l = 1, \dots, n, \quad j \neq k, j \neq l, l \neq k \quad (2.21)$$

$$x_{jj} = 0 \quad j = 1, \dots, n, \quad (2.22)$$

$$x_{jk} \in \{0,1\} \quad j, k = 1, \dots, n, \quad (2.23)$$

- Restrições disjuntivas (*Disjunctive constraints*): finalmente, na modelagem por restrições disjuntivas, seja s_j o instante de início de uma ordem j (variável real) e p_j seu

tempo de processamento. S_{jk} é uma variável binária que vale 1 se a ordem j é alocada antes de k e 0, caso contrário. As restrições que modelam as relações de precedência são mostradas em (2.24)-(2.27). Em (2.24) e (2.25) garante-se de maneira disjuntiva (ou seja, com um parâmetro M grande suficiente para que uma das restrições seja sempre verdadeira de acordo com S_{jk}) que o instante de início de uma ordem que precede a outra diste pelo menos de seu tempo de processamento do início desta outra. A restrição (2.26) mostra que S_{jk} é binária, enquanto (2.27) mostra que s_j é real;

$$s_j + p_j \leq s_k + M(1 - S_{jk}) \quad j < k = 1, \dots, n, \quad (2.24)$$

$$s_k + p_k \leq s_j + MS_{jk} \quad j < k = 1, \dots, n, \quad (2.25)$$

$$S_{jk} \in \{0,1\} \quad j < k = 1, \dots, n, \quad (2.26)$$

$$s_j \in \mathbb{R}^+ \quad j = 1, \dots, n \quad (2.27)$$

O primeiro modelo apresenta cognição bastante simples, porém tem como inconveniente um alto número de variáveis binárias, sobretudo para grandes limitantes superiores de *makespan*, ou mesmo para tempos com grande precisão (que precisam ser discretizados neste caso). Já o segundo modelo tem como desvantagem não modelar tempos ociosos entre ordens, de forma que é interessante para objetivos ligados a horizonte de programação, mas não para penalidades por atrasos e adiantamentos, por exemplo. Uma possível vantagem é a facilidade para modelar precedências arbitrárias (basta fazer $x_{jk} = 1$). Por fim, a modelagem por variáveis disjuntivas apresenta como vantagem a utilização de instantes de início reais, o que lida melhor com tempos mais precisos, além de também comportar a inserção de ociosidades entre ordens.

2.2.3.2. Lower bounds

Uma primeira ideia de *lower bound* para o problema de minimização pode ser baseada na constatação de Kim e Yano (1992). Se existem conflitos de atribuição temporal em uma programação em que o instante de término (C_j) e o prazo (d_j) são iguais, chamando de t_j o tempo em que a ordem j permanece sobreposta temporalmente, então temos como limitante inferior para uma instância D com o objetivo de minimizar a soma das penalidades por atraso e adiantamento:

$$OPT(D) \geq \sum_{j \geq 2} (j - 2)t_j$$

A partir desta constatação são desenvolvidos limitantes inferiores com base neste princípios para sequências parciais de ordens programadas. Schaller (2007) melhora os *lower bounds* incrementalmente, testando-os como parte de um algoritmo de *branch-and-bound*. Com isso, mostram obter melhores resultados.

Sourd (2004) estuda o que chama de problemas de atribuição contínua, utilizando como exemplo o problema de programação com funções de custos irregulares (como é o caso das penalidades por atraso e adiantamento ponderadas, objetos deste estudo).

2.2.3.3. Métodos exatos

Para o problema em ambiente de máquina única, Sourd (2005) apresenta um método de otimização por programação dinâmica para programar no tempo uma sequência previamente fornecida. Para este uso específico, apresenta esforço computacional bastante baixo, mesmo para instâncias de grande porte (1500 a 2000 ordens).

Sourd e Kedad-Sidhoum (2007) propõem um algoritmo *branch-and-bound* que obteve soluções ótimas para instâncias de até 50 tarefas em termos computacionais inferiores a 1000 segundos. O limitante inferior foi obtido através de uma relaxação Lagrangeana da formulação em tempos discretos (o limitante superior para o *makespan* foi a maior data de

entrega acrescida da somatória dos tempos de processamento de todas as ordens). Cada ramificação corresponde à alocação de uma ordem, ou seja, na profundidade k , existe uma sequência inicial de k tarefas alocadas e $n - k$ tarefas a serem alocadas.

Um algoritmo exato eficiente é proposto em Tanaka e Fujikuma (2008), possibilitando resolver de maneira otimizante instâncias de até 200 ordens em tempos computacionais inferiores a 15 minutos para o problema de minimização de penalidades por atraso e adiantamento. O algoritmo se baseia na geração de limitantes inferiores por programação dinâmica da relaxação Lagrangeana do problema original, sendo sucessivamente melhorado através da imposição de restrições adicionais.

Finalmente, Sourd (2009) melhora os algoritmos de *branch-and-bound* anteriores através de um *lower bound* também obtido pela relaxação Lagrangeana da formulação indexada no tempo, porém adicionando cortes neste modelo de forma que este pode ser resolvido analogamente ao caminho mais curto em um grafo. Estes cortes são feitos com base em regras de dominância, em que são testadas trocas de posições entre pares de tarefas, gerando restrições adicionais que diminuem o combinatório de soluções viável. As mesmas instâncias que Sourd e Kedad-Sidhoum (2007) resolvem em até 1000 segundos, aqui são resolvidas em até 13 segundos, apesar de haver um maior esforço computacional no cálculo dos *lower bounds*.

2.2.3.4. Heurísticas

As soluções heurísticas mais simples aplicadas ao problema de *scheduling* são as chamadas regras de liberação ou regras de despacho (*dispatching rules*). Trata-se de indicadores relacionados à ordem e, eventualmente, a sua fila, que definem a prioridades das ordens a serem programadas (sequenciamento). Desde os trabalhos seminais de Conway (1965 apud Pinedo, 2008), diversas regras foram desenvolvidas, desde as mais simples (ver Tabela 2.1 com exemplos) até regras compostas.

Tabela 2-1 – Lista de regras de despacho simples (Fonte: Pinedo, 2008)

Regra		Critério de ordenação
SIRO	Service in Random Order	aleatória
ERD	Earliest release date	data de chegada mais cedo
EDD	Earliest due date	data de entrega mais cedo
MS	Minimum slack	Menor folga entre data de entrega e tempo de processamento
SPT	Shortest Processing Time	Menor tempo de processamento
WSPT	Weighed Shortest Processing Time	Menor tempo de processamento ponderado
LPT	Longest Processing Time	Maior tempo de processamento
SST	Shortest Setup Time	Menor tempo de setup
SQ	Shortest Queue	Menor fila
SQNO	Shortest Queue at the next Operation	Menor fila na próxima operação

As regras de despacho compostas consideram mais de uma regra de despacho em um mesmo indicador de ordenação. Um exemplo é a regra ATC (*Apparent Tardiness Cost*, ou seja, custo de atraso aparente), que combina a regra WSPT e a regra MS. O indicador de ordenação das tarefas nesta regra é:

$$I_j(t) = \frac{w_j}{p_j} e^{\left(-\frac{\max(0, d_j - p_j - t)}{K\bar{p}}\right)}$$

Neste caso K é um parâmetro a ser calibrado de acordo com a importância de cada regra e $\bar{p}$ é a média dos tempos de processamento da instância.

No campo das metaheurísticas, Lee e Choi (1995) trabalharam com o problema de penalidades por atraso e adiantamento através de algoritmos genéticos, sendo um cromossomo uma sequência de tarefas com sinais de menos em momentos de tempos ociosos. A população inicial é constituída com regras de despacho; a reprodução é dada pela repetição de uma certa quantidade de soluções mais aptas; o *crossover* se dá pela permutação de partes dos cromossomos; e a mutação pela troca de posição de duas ordens.

Wan e Yen (2002) desenvolveram uma abordagem com a metaheurística *Tabu Search* para o problema, sendo a vizinhança considerada a troca de pares adjacentes. A lista tabu foi parametrizada nos 7 últimos movimentos; o critério de aspiração é a busca por uma solução com melhor função objetivo (menor penalidade).

M'Hallah (2007) aplica uma metaheurística híbrida, gerando uma população de soluções através de um algoritmo genético, e, posteriormente, utiliza os métodos de *Hill Climbing* e *Simulated Annealing* para fazer uma busca estendida na vizinhança.

Valente e Moreira (2009) utilizam o método GRASP (*Greedy Randomized Adaptive Search Procedure*) em regras de despacho, utilizando uma lista restrita de candidatos (ordens) que recebem um viés exponencial de escolha de acordo com sua prioridade pela regra em questão.

2.3. Abordagens conjuntas entre programação e operações de corte

A literatura a respeito de corte e empacotamento é bastante extensa e inúmeros métodos foram desenvolvidos para diferentes tipo de problemas, como visto até agora. Embora os modelos satisfaçam as restrições de demanda e possibilitem a minimização do desperdício de material para diferentes situações, os padrões de corte gerados interagem em outros níveis de decisão da indústria, como é o caso da programação.

2.3.1. Problema das pilhas abertas

A grande maioria das abordagens conjuntas na literatura se preocupa com o problema de estoque gerado e seu armazenamento, em classes de problemas denominadas por Yanasse (1997) de MOSP (*minimization of open stack problem*) e MORP (*minimization of order spread problem*). Este último é demonstrado ser NP-difícil por Fink e Voß (1999), ainda que Foerster e Wäscher (1998) o classifique como NP-completo por ser uma generalização do problema do caixeiro viajante. O primeiro também é demonstrado ser NP-difícil por Linhares e Yanasse (2002).

Segundo Pileggi, Morabito e Arenales (2007), uma pilha é chamada de “aberta” se a manufatura da ordem em questão foi começada, porém não finalizada – o que só ocorre quando após o processamento do último item desta. Uma vez que os lotes de peças podem estar segmentados em inúmeros padrões de corte, as dificuldades de espaço e fluxo no processo são trazidas à tona. Yanasse (1997) aponta como inconvenientes de ordens inacabadas o alto custo de estoque, manipulação de materiais, potenciais atrasos em entregas (e, possivelmente, de faturamento) ou custos de transporte pela dispersão dos lotes.

Alves e Carvalho (2008) apresentam uma formulação para o problema de minimização de pilhas abertas e desenvolvem um método exato baseado em *branch-and-bound*.

Tal abordagem contempla um dos três objetivos da programação que é o de velocidade de fluxo/estoque intermediário. Como o problema de corte e empacotamento naturalmente abarca o objetivo de utilização dos recursos produtivos, o objetivo aqui negligenciado é justamente o de atendimento de datas.

2.3.2. Dimensionamento de lotes

Outro problema que já foi abordado conjuntamente ao problema de corte de estoque é o de dimensionamento de lotes (PDL). Segundo Poldi (2007), o PDL é um problema de programação voltado para a definição das quantidades a serem produzidas em cada período em um dado horizonte de planejamento.

Modelos de programação linear para o problema acoplado de dimensionamento de lotes com o de corte unidimensional são propostos em Poltronieri (2006) na indústria papaleira; e de corte bidimensional em Gramani (2001) na indústria moveleira. Poldi (2007) propõe um método de geração de colunas para o problema acoplado em ambas dimensões.

2.3.3. Atendimento de datas

O problema de atendimento de datas, escopo de pesquisa deste trabalho, ainda carece de propostas de modelos e abordagens.

Reinertsen e Vossen (2009) incluem em seu modelo matemático a questão das datas de entrega (*due dates*), de forma que os elementos de programação são considerados. Aqui, o objetivo ainda é minimizar o uso de materiais (ocupação), porém o respeito aos prazos está na forma de restrições. Para refletir a flexibilidade de permitir eventuais atrasos, insere-se um parâmetro adicional que é o número máximo de itens que podem desrespeitar a data de entrega em cada período (devidamente penalizados na função objetivo).

Os autores propõe um método de geração de colunas – muito utilizado em problemas de corte e empacotamento – que consiste na adição de vetores de padrões de corte a cada iteração do modelo de programação linear.

Chrysosolouris, Papakostas e Mourtzis (2011) também retratam o problema de programação de operações de corte, porém neste caso de objetos irregulares (corte do tipo *nesting*). Nesta caso, a modelagem matemática não é explicitada, porém a heurística proposta em 4 fases considera o atendimento a prazos como uma restrição.

2.3.4. Outras abordagens

O problema de *scheduling* com formação de lotes pode ser encontrado amplamente na literatura, entretanto não retrata as restrições de dimensionalidade e geometria dos itens. Potts e Kovalyov (2000) apresentam uma boa revisão deste tipo de problema de programação. Entretanto, como mencionado, neste caso considera-se produções com lotes limitados pelo número de unidades (*b*) em cada batelada. Este caso não pode ser adaptado para problemas de corte, exceto quando os itens são idênticos entre si.

Por fim, Ragsdale e Zobel (2004) abordam o problema do corte de itens quando existe precedência de corte, o que é chamado de corte ordenado. Não existe menção a atendimento de datas, apenas ao respeito às restrições de precedência entre ordens.

2.4. Considerações finais

Da revisão que será apresentada, nota-se que as duas primeiras partes são muito ricas: mesmo para cada subproblema da área encontra-se vasta e rica literatura, com diversos trabalhos publicados recentemente e com grande perspectiva de continuidade. Por outro lado, as abordagens conjuntas ainda são em menor número e, claramente, não apresentam uma unidade em torno de um mesmo problema, ainda que haja grande complexidade teórica e interesse prático.

3. MODELAGEM MATEMÁTICA PARA O PROBLEMA

Este capítulo tem como objetivo apresentar um modelo de programação linear inteira mista que retrata o problema descrito na seção 2 à luz da literatura apresentada na seção 3. Para isso, inicialmente serão mostrados dois modelos de programação linear inteira mista da literatura que retratam os problemas de *scheduling* e *bin packing* separadamente. Posteriormente, é feita uma adaptação de ambos modelos para resultar em uma formulação conjunta.

4.1. Modelagens específicas

Nesta subseção apresentaremos adaptações das formulações propostas por Hifi et al. (2010) e Biskup e Feldmann (2001) respectivamente para os problemas de *bin packing* bidimensional retangular com *bins* idênticos e itens quaisquer e *scheduling* em ambiente de máquina única minimizando a soma dos atrasos e adiantamentos ponderados (atendimento a prazos).

4.1.1. Modelo para o problema de *bin packing*

A formulação proposta por Hifi et al. (2010), na verdade, presta-se a retratar o problema de *bin packing* tridimensional. Entretanto, por apresentar características interessantes como o uso de variável reais no posicionamento de itens e no posicionamento dos itens em *bins*, foi aqui adaptada para sua versão bidimensional retangular. A notação também foi adaptada para unificar o entendimento entre as três formulações desta seção, deixando parâmetros e variáveis análogos com a mesma representação. O modelo proposto contempla a situação de *bins* idênticos para qualquer conjunto de itens.

Dado um conjunto de n itens ($i = 1, 2, \dots, n$) com dimensões l_i e w_i , o objetivo do problema é empacotá-los no menor número de *bins* possível, sendo estes idênticos entre si e com dimensões L e W . O modelo se utiliza de um parâmetro que é o limite superior do número de *bins* a serem utilizados, $\bar{\gamma}$.

O posicionamento de um item é dado pelo vetor (x_i, y_i, γ_i) , em que x_i é a coordenada no eixo x de seu canto inferior esquerdo no *bin* e y_i é a coordenada no eixo y do mesmo canto.

Assume-se como (0, 0) a coordenada do canto inferior esquerdo do *bin*. Já γ_i é o rótulo do *bin* a que o item i é atribuído. A função objetivo é minimizar γ , que é o maior rótulo γ_i .

$$\min \quad \gamma \quad (3.1)$$

$$s. a. \quad X_{ij} + X_{ji} + Y_{ij} + Y_{ji} + S_{ij} + S_{ji} = 1 \quad i < j = 1, \dots, n, \quad (3.2)$$

$$x_i - x_j + L(X_{ij} - S_{ij} - S_{ji}) \leq L - l_i \quad i \neq j = 1, \dots, n, \quad (3.3)$$

$$y_i - y_j + W(Y_{ij} - S_{ij} - S_{ji}) \leq W - w_i \quad i \neq j = 1, \dots, n, \quad (3.4)$$

$$(\bar{\gamma} - 1)(X_{ij} + X_{ji} + Y_{ij} + Y_{ji}) + \\ + \gamma_i - \gamma_j + \bar{\gamma}S_{ij} \leq \bar{\gamma} - 1 \quad i \neq j = 1, \dots, n, \quad (3.5)$$

$$0 \leq x_i \leq L - l_i \quad i = 1, \dots, n, \quad (3.6)$$

$$0 \leq y_i \leq W - w_i \quad i = 1, \dots, n, \quad (3.7)$$

$$0 \leq \gamma_i \leq \gamma \leq \bar{\gamma} \quad i = 1, \dots, n, \quad (3.8)$$

$$X_{ij}, Y_{ij}, S_{ij} \in \{0,1\} \quad i \neq j = 1, \dots, n, \quad (3.9)$$

$$x_i, y_i, \gamma_i \in \mathbb{R}^+ \quad i = 1, \dots, n \quad (3.10)$$

As variáveis binárias X_{ij} têm valor 1 se o item i está à esquerda do item j (ou seja, sua coordenada x dista pelo menos de l_i da coordenada x de j) no mesmo *bin*. Analogamente, as variáveis binárias Y_{ij} têm valor 1 se o item i está abaixo do item j (ou seja, sua coordenada y dista pelo menos de w_i da coordenada y de j) no mesmo *bin*. Finalmente, S_{ij} vale 1 se o item i está em um *bin* com rótulo inferior a j , ou seja, $\gamma_i < \gamma_j$.

Analisando o modelo passo-a-passo, temos os seguintes componentes:

A função objetivo (3.1) orienta a otimização na direção de minimizar o maior rótulo de *bin* (γ) utilizado por um item i , já que a restrição (3.8) garante que $\gamma_i \leq \gamma$, $\forall i$, $i = 1, \dots, n$.

$$X_{ij} + X_{ji} + Y_{ij} + Y_{ji} + S_{ij} + S_{ji} = 1 \quad i < j = 1, \dots, n, \quad (3.2)$$

A restrição (3.2) rege a não-sobreposição entre dois itens i e j . Através dela, fica garantido que o item i está à esquerda de j ($X_{ij} = 1$) ou o item j está à esquerda de i ($X_{ji} = 1$) ou o item i está abaixo de j ($Y_{ij} = 1$) ou o item j está abaixo de i ($Y_{ji} = 1$) ou o item i está em um *bin* de rótulo inferior ao de j ($S_{ij} = 1$) ou o item j está em um *bin* de rótulo inferior ao de i ($S_{ji} = 1$).

Interessante observar que, embora fisicamente possam ocorrer duas dessas condições simultaneamente (por exemplo, o item i pode estar à esquerda e abaixo de j ao mesmo tempo), basta que uma destas condições se verifique para que seja garantida a não-sobreposição. Desta forma, a restrição pode ser uma igualdade e não uma inequação em que o lado esquerdo deve ser maior ou igual a 1. Esta última estaria correta e se aproximaria mais do sentido físico do problema, porém a primeira é suficiente para garantir que dois itens estejam separados e há ganhos em redução do espaço combinatório desta forma. Não há perda de generalidade, já que as variáveis de posicionamento ficam livres, como será visto adiante.

$$x_i - x_j + L(X_{ij} - S_{ij} - S_{ji}) \leq L - l_i \quad i \neq j = 1, \dots, n, \quad (3.3)$$

$$y_i - y_j + W(Y_{ij} - S_{ij} - S_{ji}) \leq W - w_i \quad i \neq j = 1, \dots, n, \quad (3.4)$$

A restrição (3.3) garante que o item i diste pelo menos de seu comprimento (l_i) em relação a j na coordenada x se estiverem no mesmo *bin* e i estiver à esquerda de j . Em outras palavras, a restrição (3.3) garante o sentido físico de X_{ij} . Note que se i e j não estiverem no mesmo *bin*, $X_{ij} = 0$ e, ainda, $S_{ij} = 1$ ou $S_{ji} = 1$, de forma que a inequação resultante é *dummy* ($x_i - x_j \leq 2L - l_i$). Da mesma forma, se estiverem no mesmo *bin* e $X_{ij} = 0$, também resulta uma restrição *dummy* ($x_i - x_j \leq L - l_i$). Apenas se $X_{ij} = 1$ a restrição torna-se relevante ($x_j - x_i \geq l_i$). Assim, a garantia de generalidade mencionada no parágrafo anterior fica mostrada, já que o valor $X_{ij} = 0$ não restringe as possibilidades de x_i e x_j .

A restrição (3.4) é análoga a (3.3) para o eixo y : garante o sentido físico de $Y_{ij} = 1$, ao impor que o item i diste pelo menos de sua largura (w_i) em relação ao item j na coordenada y .

$$(\bar{\gamma} - 1)(X_{ij} + X_{ji} + Y_{ij} + Y_{ji}) + \gamma_i - \gamma_j + \bar{\gamma}S_{ij} \leq \bar{\gamma} - 1 \quad i \neq j = 1, \dots, n, \quad (3.5)$$

A restrição (3.5) garante sentido físico na situação em que $S_{ij} = 1$. Note que se isto acontece, pela restrição (3.2) implica-se que a soma $X_{ij} + X_{ji} + Y_{ij} + Y_{ji}$ deve ser 0. Desta forma, a restrição (3.5) toma a forma $\gamma_j - \gamma_i \geq 1$. Assim, garante-se que o rótulo do *bin* em que está o item i seja pelo menos 1 unidade menor do que o rótulo do bin em que está o item j .

Veja que, se estamos em uma situação em que os itens i e j estão no mesmo *bin* ($S_{ij} = S_{ji} = 0$), então a soma $X_{ij} + X_{ji} + Y_{ij} + Y_{ji}$ deve ser 1, de forma que a restrição que resulta é $\gamma_i - \gamma_j \leq 0$. Como esse conjunto de restrições vale para todo par i, j com $i \neq j$, então também valerá que $\gamma_j - \gamma_i \leq 0$, de tal forma que os rótulos γ_i e γ_j necessariamente serão iguais.

Outro ponto interessante desta restrição é que, já que itens posicionados em *bins* diferentes devem ter seus rótulos distando em valor de pelo menos 1 unidade e a função objetivo é minimizar o maior rótulo, decorre que os valores γ_i serão necessariamente inteiros, mesmo que isto não seja imposto como restrição. Trata-se de outro artifício que deixa a modelagem computacionalmente mais interessante.

$$0 \leq x_i \leq L - l_i \quad i = 1, \dots, n, \quad (3.6)$$

$$0 \leq y_i \leq W - w_i \quad i = 1, \dots, n, \quad (3.7)$$

As restrições (3.6) e (3.7) garantem que os itens estejam posicionados integralmente no interior do *bin*, ou seja, as coordenadas de seus cantos inferiores-esquerdos devem ser superiores ou iguais a 0 e inferiores ou iguais aos limites do bin descontados da dimensão do próprio item. Isto é está retratado em (3.6) para o comprimento (coordenada x) e em (3.7) para a largura (coordenada y).

$$0 \leq \gamma_i \leq \gamma \leq \bar{\gamma} \quad i = 1, \dots, n, \quad (3.8)$$

A restrição (3.8) faz papel análogo a (3.6) e (3.7) para as variáveis de rótulo dos *bins*. Nela garante-se apenas a definição de que γ é o maior rótulo de *bin* e $\bar{\gamma}$ é um limitante superior para o número de *bins* utilizados.

As declarações (3.9) e (3.10) apenas definem as variáveis de decisão utilizadas como binárias e reais, respectivamente.

4.1.2. Modelo para o problema *scheduling*

A formulação proposta por Biskup e Feldman (2001) originalmente contempla o problema de datas de entrega comuns. A adaptação desta particularidade para o caso mais genérico de datas de entrega particulares a cada item é bastante simples, como a seguir apresentado. A escolha deste modelo como base se deu por (i) se tratar de uma formulação com o mesmo ambiente estudado deste trabalho (máquina única, $\alpha = 1$); (ii) com a mesma função objetivo (soma de atrasos e adiantamentos ponderados, $\gamma = \sum \alpha_i E_i + \beta_i T_i$) no que tange à programação; (iii) e com a mesma tônica explorar as dimensões contínuas como variáveis reais (pois trata-se de um modelo com restrições disjuntivas em que o tempo é tratado de maneira contínua).

O modelo apresentado a seguir é uma adaptação da formulação de Biskup e Feldman (2001). Além da adaptação para o caso mais genérico de datas de entrega específicas por item, a notação também foi adaptada, como na subseção 4.1.1, na tentativa de aproximar termos com significados análogos.

Dado um conjunto de n ordens de produção ($i = 1, 2, \dots, n$) com tempo de processamento p_i , data de entrega d_i , penalidade por tempo de término em adiantamento α_i e penalidade por término de término em atraso β_i ; o objetivo do problema é programá-los de forma a minimizar a soma dos atrasos e adiantamentos ponderados pelas penalidades de cada ordem.

O instante de início do processamento da ordem i é dado pela variável de decisão s_i , seu tempo de adiantamento por E_i e seu tempo em atraso por T_i .

$$\min \quad \sum_{i=1}^n \alpha_i E_i + \sum_{i=1}^n \beta_i T_i \quad (3.11)$$

$$s.a. \quad T_i \geq s_i + p_i - d_i \quad i = 1, \dots, n, \quad (3.12)$$

$$E_i \geq d_i - s_i - p_i \quad i = 1, \dots, n, \quad (3.13)$$

$$s_i + p_i \leq s_j + M(1 - S_{ij}) \quad i < j = 1, \dots, n, \quad (3.14)$$

$$s_j + p_j \leq s_i + MS_{ij} \quad i < j = 1, \dots, n, \quad (3.15)$$

$$S_{ij} \in \{0,1\} \quad i < j = 1, \dots, n, \quad (3.16)$$

$$T_i, E_i, s_i \in \mathbb{R}^+ \quad i = 1, \dots, n \quad (3.17)$$

As variáveis binárias S_{ij} , diferente porém analogamente ao modelo anterior, têm valor 1 se a ordem i está programada anteriormente à ordem j (ou seja, o instante de início de i dista de pelo menos p_i do tempo de início de j , que é seu tempo de processamento).

Analisando o modelo passo-a-passo, temos os seguintes componentes:

$$\min \quad \sum_{i=1}^n \alpha_i E_i + \sum_{i=1}^n \beta_i T_i \quad (3.11)$$

A função objetivo (3.11) orienta a otimização na direção de minimizar a soma dos adiantamentos das ordens (E_i) ponderados por suas respectivas penalidades (α_i), assim como a soma dos atrasos das ordens (T_i) ponderados por suas respectivas penalidades (β_i).

$$T_i \geq s_i + p_i - d_i \quad i = 1, \dots, n, \quad (3.12)$$

$$E_i \geq d_i - s_i - p_i \quad i = 1, \dots, n, \quad (3.13)$$

A restrição (3.12) retrata a definição de da variável T_i . Sendo T_i o atraso da ordem i , deve ser nulo quando a ordem tiver seu instante de término antes da data de entrega, ou tomar o valor da diferença entre a data de entrega e o instante de término quando aquela for anterior a este.

Uma vez que a função objetivo tem direção de minimização e a variável T_i impacta positivamente nela, ao impor que seu valor deve ser superior ou igual à diferença entre seu instante de término ($s_i + p_i$) e sua data de entrega (d_i), naturalmente seu valor será exatamente essa diferença. Por outro lado, se a diferença for negativa, prepondera a restrição de não-negatividade (3.17) e T_i assume valor nulo.

A restrição (3.13) é análoga para a definição de adiantamento, garantindo que E_i seja nula quando a ordem é programada em atraso; e seja a diferença entre a data de entrega (d_i) e o instante de término ($s_i + p_i$) quando esta for positiva (ordem programada em adiantamento).

$$s_i + p_i \leq s_j + M(1 - S_{ij}) \quad i < j = 1, \dots, n, \quad (3.14)$$

$$s_j + p_j \leq s_i + MS_{ij} \quad i < j = 1, \dots, n, \quad (3.15)$$

As restrições (3.14) e (3.15) são as chamadas restrições disjuntivas, que garantem a não-sobreposição temporal entre a execução de duas ordens. Se a ordem i for programada antes da ordem j ($S_{ij} = 1$), então (3.14) garante esta não sobreposição, já que resta a inequação $s_i + p_i \leq s_j$. Por outro lado, neste caso, a restrição (3.15) torna-se *dummy*, já que M é um número suficientemente grande de modo que $M + s_i$ seja sempre maior que $s_j + p_j$. De maneira análoga, se a ordem i for programada depois da ordem j ($S_{ij} = 0$), então é a restrição (3.14) que se torna *dummy*, enquanto (3.15) garante que o instante de início da ordem i (s_i) diste pelo menos do tempo de processamento de j (p_j) do instante de início desta última (s_j). Em outras palavras, garante que $s_j + p_j \leq s_i$.

Finalmente, as declarações (3.16) e (3.17) apenas definem as variáveis de decisão utilizadas como binárias e reais, respectivamente.

4.2. Modelagens para o problema de programação de operações de corte

Apresentadas as formulações de *bin packing* e *scheduling* que serviram de base para o desenvolvimento do modelo unificado de programação de operações de corte, nesta subseção este será apresentado.

O modelo de programação linear inteira mista deste trabalho utiliza a mesma lógica das formulações anteriormente apresentadas para garantir não sobreposição física (corte) e temporal (programação) dos itens a serem programados. As devidas adaptações foram realizadas, sobretudo com o intuito de alinhar a questão de que a não-sobreposição temporal já não concerne apenas o tempo de processamento do item, mas sim do lote (*bin*) a ser cortado.

Seja um conjunto de n itens retangulares a serem programados em uma operação de corte ($i = 1, 2, \dots, n$) com tempo de processamento p_i , data de entrega d_i , penalidade por tempo de término em adiantamento α_i , penalidade por término em atraso β_i , dimensões l_i (comprimento) e w_i (largura). O objetivo do problema é programá-los de forma a minimizar a soma das penalidades por adiantamentos, atrasos e uso de matéria-prima, ponderados por seus respectivos pesos/custos. Sua programação implica na formação de lotes a serem cortados em *bins* idênticos de comprimento L , largura W , custo θ e tempo de preparação s_t .

O instante de início do processamento do *bin* em que está empacotado o item i é dado por s_i . O tempo de processamento deste *bin* em que o item i está é dado pela variável t_i . Note que t_i é uma variável, pois o tempo de processamento depende de quais itens são empacotados em determinado *bin*. Seu adiantamento e atraso em relação a sua data de entrega são representados respectivamente pelas variáveis E_i e T_i . O instante de início do processamento da ordem i é dado pela variável de decisão s_i , seu tempo de adiantamento por E_i e seu tempo em atraso por T_i . Finalmente, as coordenadas x e y do canto inferior esquerdo do item i no *bin* em que está posicionado são dados, respectivamente, pelas variáveis x_i e y_i , tendo o canto inferior esquerdo do *bin* as coordenadas $(0, 0)$.

Novamente, utilizamos o conjunto de variáveis binárias X_{ij} , com valor 1 se o item i está à esquerda do item j (ou seja, sua coordenada x dista pelo menos de l_i da coordenada x de j) no mesmo *bin*. Analogamente, quando o item i está abaixo do item j (ou seja, sua coordenada y dista pelo menos de w_i da coordenada y de j) no mesmo *bin*, as variáveis binárias Y_{ij} têm valor 1. Seus significados, portanto, são idênticos ao modelo de Hifi et al. (2010). Já a variável S_{ij} , também análoga, ganha valor 1 se o item i está em um *bin* programado anteriormente ao *bin* em que está o item j .

Por fim, b_i é uma variável auxiliar que surge como um subterfúgio matemático para contar o número de *bins* a serem utilizados sem fazer uso de mais variáveis binárias, que sobrecarregariam (ainda mais) o modelo do ponto de vista computacional. Esta variável,

embora real, só toma os valores 0 e 1 devido à restrição em que aparece, tendo valor 1 quando o item i “representa” o *bin* em que está e 0 caso contrário (apenas 1 item “representa” cada *bin*).

A seguir, mostra-se um resumo dos parâmetros e variáveis explicados anteriormente e, então, em (3.18)-(3.31) está o modelo matemático para o problema proposto.

Parâmetros:

θ = custo de um bin
 L = comprimento do bin
 W = largura do bin
 st = tempo de preparação da máquina para receber um lote
 α_i = custo por unidade de adiantamento do item i
 β_i = custo por unidade de atraso do item i
 d_i = data de entrega do item i
 p_i = tempo de processamento do item i
 l_i = comprimento do item i
 w_i = largura do item i

Variáveis:

s_i = instante de início do item i
 E_i = adiantamento do item i
 T_i = atraso do item i
 t_i = tempo de processamento do **bin em que está** o item i
 x_i = coordenada x do canto inferior esquerdo do item i
 y_i = coordenada y do canto inferior esquerdo do item i
 $X_{ij} = 1$, se o item i precede o item j no sentido x ;
0 caso contrário
 $Y_{ij} = 1$, se o item i precede o item j no sentido y ;
0 caso contrário
 $S_{ij} = 1$, se o item i precede o item j temporalmente (*bin* programado mais cedo);
0 caso contrário

$b_j = 1$ se o item j “representa” um *bin* usado,
 0 caso contrário

$$\min \quad \sum_{i=1}^n \alpha_i E_i + \sum_{i=1}^n \beta_i T_i + \theta \sum_{i=1}^n b_i \quad (3.18)$$

$$s. a. \quad T_i \geq s_i + t_i - d_i \quad i = 1, \dots, n, \quad (3.19)$$

$$E_i \geq d_i - s_i - t_i \quad i = 1, \dots, n, \quad (3.20)$$

$$s_i + t_i \leq s_j + M(1 - S_{ij}) \quad i \neq j = 1, \dots, n, \quad (3.21)$$

$$s_i \leq s_j + MS_{ji} \quad i \neq j = 1, \dots, n, \quad (3.22)$$

$$t_i = st + p_i + \sum_{\substack{j=1 \\ j \neq i}}^n ((1 - S_{ij} - S_{ji})(p_j)) \quad i = 1, \dots, n, \quad (3.23)$$

$$x_i - x_j + L(X_{ij} - S_{ij} - S_{ji}) \leq L - l_i \quad i \neq j = 1, \dots, n, \quad (3.24)$$

$$y_i - y_j + W(Y_{ij} - S_{ij} - S_{ji}) \leq W - w_i \quad i \neq j = 1, \dots, n, \quad (3.25)$$

$$X_{ij} + X_{ji} + Y_{ij} + Y_{ji} + S_{ij} + S_{ji} = 1 \quad i > j = 1, \dots, n, \quad (3.26)$$

$$b_i \geq 1 - \sum_{j=1}^{i-1} (1 - S_{ij} - S_{ji}) \quad i = 1, \dots, n, \quad (3.27)$$

$$0 \leq x_i \leq L - l_i \quad i = 1, \dots, n, \quad (3.28)$$

$$0 \leq y_i \leq W - w_i \quad i = 1, \dots, n, \quad (3.29)$$

$$x_i, y_i, b_i, s_i, T_i, E_i, t_i \in \mathbb{R}^+ \quad i = 1, \dots, n, \quad (3.30)$$

$$X_{ij}, Y_{ij}, S_{ij} \in \{0,1\} \quad i \neq j = 1, \dots, n \quad (3.31)$$

Analisando o modelo passo-a-passo, temos os seguintes componentes:

$$\min \quad \sum_{i=1}^n \alpha_i E_i + \sum_{i=1}^n \beta_i T_i + \theta \sum_{i=1}^n b_i \quad (3.18)$$

A função objetivo (3.18) orienta a otimização na direção de minimizar três penalidades: (i) a soma dos adiantamentos das ordens (E_i) ponderados por suas respectivas penalidades (α_i); (ii) a soma dos atrasos das ordens soma dos atrasos das ordens (T_i) ponderados por suas respectivas penalidades (β_i); e (iii) a soma do número de *bins* utilizados multiplicada pela penalidade unitária de um deles (custo). Desta forma, esta função objetivo retoma a noção de contemplar ao mesmo tempo dois objetivos comuns a sistemas de produção com operações de corte: o atendimento de datas de entrega e o bom uso de matérias-primas.

$$T_i \geq s_i + t_i - d_i \quad i = 1, \dots, n, \quad (3.19)$$

$$E_i \geq d_i - s_i - t_i \quad i = 1, \dots, n, \quad (3.20)$$

As restrições (3.19) e (3.20) são análogas à formulação de Biskup e Feldmann (2001): garantem a coerência das variáveis T_i e E_i com seus respectivos sentidos de atraso e adiantamento. O único destaque está na diferença do tempo de processamento, que na formulação da subseção 4.1.2 é uma propriedade do item, enquanto aqui este tempo é uma propriedade do *bin* em que está o item. O tempo t_i , definido com mais detalhes adiante na restrição (6), substitui o tempo de processamento do item, já que o instante de término do item só ocorre quando o *bin* a que pertence é retirado da máquina. Deve, portanto, contemplar o processamento de todos os itens deste lote.

$$s_i + t_i \leq s_j + M(1 - S_{ij}) \quad i \neq j = 1, \dots, n, \quad (3.21)$$

$$s_i \geq s_j + MS_{ji} \quad i \neq j = 1, \dots, n, \quad (3.22)$$

Em (3.21) e (3.22) temos as restrições disjuntivas que garantem a coerência temporal entre os itens programados. Diferentemente da formulação da subseção 4.1.2, não se trata de garantir uma não-sobreposição dos itens de maneira coerente com S_{ij} . Aqui, três situações distintas

devem ser contempladas: (i) o item i pode pertencer a um *bin* que precede o *bin* em que está j na programação ($S_{ij} = 1$ e $S_{ji} = 0$), de forma que j só pode ser programado após o instante de término de i ; (ii) o item i pode pertencer a um *bin* que sucede aquele em que está j ($S_{ij} = 0$ e $S_{ji} = 1$), devendo o item i ter instante de início posterior ao instante de término do *bin* em que está j ; (iii) e os itens i e j podem pertencer ao mesmo *bin* ($S_{ij} = 0$ e $S_{ji} = 0$), devendo ter seus instantes de término idênticos.

As restrições (3.21) e (3.22) proporcionam estas três restrições:

- No primeiro caso ($S_{ij} = 1$ e $S_{ji} = 0$), da restrição (3.21) resulta que $s_i + t_i \leq s_j$, de forma que o instante de início do *bin* em que está o item j (s_j) é maior ou igual ao instante de término do *bin* em que está o item i ($s_i + t_i$). A restrição (3.22), neste caso ($S_{ji} = 0$), resulta $s_j \geq s_i$, tornando-se irrelevante por ser menos restritiva do que a anterior;
- No segundo caso ($S_{ij} = 0$ e $S_{ji} = 1$), ocorre algo análogo: da restrição (3.21) resulta que $s_j + t_j \leq s_i$ e da restrição (3.22) resulta que $s_i \geq s_j$ (que, novamente, se torna irrelevante perante a primeira), fazendo com que o instante de início do *bin* em que está o item i (s_i) seja maior ou igual ao instante de término do *bin* em que está o item j ($s_j + t_j$);
- No terceiro caso ($S_{ij} = 0$ e $S_{ji} = 0$) a restrição (3.21) torna-se *dummy* tanto para i quanto para j e da restrição (3.22) resulta $s_i \geq s_j$ e $s_j \geq s_i$, ou seja, garante-se que $s_j = s_i$, como desejado.

O Quadro 3.1 resume ilustrativamente os cenários de S_{ij} e S_{ji} .

$$t_i = st + p_i + \sum_{\substack{j=1 \\ j \neq i}}^n ((1 - S_{ij} - S_{ji})(p_j)) \quad i = 1, \dots, n, \quad (3.23)$$

Na restrição (3.23) é definida a variável real t_i , que representa o tempo de processamento do bin em que o item i está contido. O tempo de produção deste lote é dado pela soma do tempo de preparação (st) e os tempos de processamento dos itens que estão no mesmo bin que o item i . Veja que isto é feito sem a necessidade de novas variáveis binárias, já que só ocorre que dois itens i e j estejam no mesmo bin se $S_{ij} = 0$ e $S_{ji} = 0$. Assim, para parcela da somatória é o

produto do tempo de processamento de cada item pela sua pertinência ou não no bin. Em outras palavras, o termo $1 - S_{ij} - S_{ji}$ provocará a soma de uma parcela nula quando o item j não pertencer ao mesmo bin que i ; ou uma parcela de p_j caso contrário. Como a variável binária S_{ij} não está definida para $i = j$, o tempo de processamento do próprio item i é somado previamente.

	$S_{ij} = 0$	$S_{ij} = 1$
$S_{ji} = 0$	$s_i + t_i \leq s_j + M$ (dummy) $s_j + t_j \leq s_i + M$ (dummy) $s_i \leq s_j$ $s_j \leq s_i$ Portanto, $s_i = s_j$ (mesma placa)	$s_i + t_i \leq s_j$ $s_j + t_j \leq s_i + M$ (dummy) $s_i \leq s_j + M$ (dummy) $s_i \leq s_j$ (redundante) Portanto, $s_i + t_i \leq s_j$
$S_{ji} = 1$	$s_i + t_i \leq s_j + M$ (dummy) $s_j + t_j \leq s_i$ $s_j \leq s_i$ (redundante) $s_j \leq s_i + M$ (dummy) Portanto, $s_j + t_j \leq s_i$	Inviável pela restrição (3.26)

Quadro 3.1 – Cenários de S_{ij} e S_{ji} e suas consequências na formulação

Importante notar que esta forma de cálculo é uma simplificação da realidade, já que o tempo de processamento em uma operação de corte não é uma soma de tempos de processamentos unitários. Entretanto, pode-se considerar um bom modelo da realidade, já que atende à lógica de aumentar o tempo de processamento conforme aumenta o número de itens no *bin* e permite retratar a complexidade individual de um determinado item em relação a outro.

$$x_i - x_j + L(X_{ij} - S_{ij} - S_{ji}) \leq L - l_i \quad i \neq j = 1, \dots, n, \quad (3.24)$$

$$y_i - y_j + W(Y_{ij} - S_{ij} - S_{ji}) \leq W - w_i \quad i \neq j = 1, \dots, n, \quad (3.25)$$

As restrições (3.24) e (3.25) são idênticas (inclusive em notação) às restrições (3.3) e (3.4) do modelo apresentado de Hifi et al. (2010) na subseção 4.1.1. Garantem, portanto, a não-sobreposição, respectivamente nos eixos x e y , de itens que estão em um mesmo *bin*.

$$X_{ij} + X_{ji} + Y_{ij} + Y_{ji} + S_{ij} + S_{ji} = 1 \quad i > j = 1, \dots, n, \quad (3.26)$$

A restrição (3.26), que também é idêntica à restrição (3.2) no modelo apresentado na subseção 4.1.1, garante que para qualquer par de itens i e j : o item i está à esquerda de j ($X_{ij} = 1$) ou o item j está à esquerda de i ($X_{ji} = 1$) ou o item i está abaixo de j ($Y_{ij} = 1$) ou o item j está abaixo de i ($Y_{ji} = 1$) ou o item i está em um *bin* de rótulo inferior a j ($S_{ij} = 1$) ou o item j está em um *bin* de rótulo inferior a i ($S_{ji} = 1$).

Novamente, chama-se a atenção para o fato de que, embora fisicamente possam ocorrer duas dessas condições simultaneamente, é suficiente que apenas uma destas condições se verifique para que seja garantida a não-sobreposição do par i, j . Assim, a restrição pode ser uma igualdade sem perda de generalidade.

$$b_i \geq 1 - \sum_{j=1}^{i-1} (1 - S_{ij} - S_{ji}) \quad i = 1, \dots, n, \quad (3.27)$$

A restrição (3.27) permite a contagem do número de *bins* utilizados na solução obtida, que é feita na função objetivo.

Perceba que a variável b_i subtrai de uma unidade o número de itens com índice inferior a i que estão no mesmo *bin* que o item i ($S_{ij} = S_{ji} = 0 \Rightarrow 1 - S_{ij} - S_{ji} = 1$). Como decorrência, se existe algum item nessas condições, a variável assumirá valor nulo, já que o resultado do lado direito da inequação será não-positivo e, pela definição da variável $b_i \in \mathbb{R}^+$, o valor mais conveniente (levando em conta que b_i penaliza a função objetivo) é 0. Por outro lado, se nenhum dos itens $j < i$, pertencer ao mesmo *bin*, o lado direito da inequação será 1 e o valor

mais conveniente e viável para a variável b_i é 1. Assim, este item estaria “representando” o *bin* em que está.

O quadro 3.2 resume ilustrativamente a ideia, exemplificando com um caso em que os itens 1, 3 e 5 estão em um *bin*, enquanto os itens 2 e 4 estão em outro, sendo que o o *bin* dos itens ímpares foi programado posteriormente ao dos itens pares.

	j = 1	j = 2	j = 3	j = 4	j = 5	b_i
i = 1						$b_1 \geq 1$
i = 2	$S_{ij} = 1$ $S_{ji} = 0$					$b_2 \geq 1 - (1 - 1 - 0)$ $b_2 \geq 1$
i = 3	$S_{ij} = 0$ $S_{ji} = 0$	$S_{ij} = 0$ $S_{ji} = 1$				$b_3 \geq 1 - (1 - 0 - 0) - (1 - 0 - 1)$ $b_3 \geq 0$
i = 4	$S_{ij} = 1$ $S_{ji} = 0$	$S_{ij} = 0$ $S_{ji} = 0$	$S_{ij} = 1$ $S_{ji} = 0$			$b_4 \geq 1 - (1 - 1 - 0) - (1 - 0 - 0) - (1 - 1 - 0)$ $b_4 \geq 0$
i = 5	$S_{ij} = 0$ $S_{ji} = 0$	$S_{ij} = 0$ $S_{ji} = 1$	$S_{ij} = 0$ $S_{ji} = 0$	$S_{ij} = 0$ $S_{ji} = 1$		$b_5 \geq 1 - (1 - 0 - 0) - (1 - 0 - 1) - (1 - 0 - 0) - (1 - 0 - 1)$ $b_5 \geq -1$

Quadro 3.2 – Exemplo de computação de b_i em que em um *bin* são alocados os itens 2 e 4 e outro, programado posteriormente, os itens 1, 3 e 5

Desta forma, em cada conjunto de itens que pertencem a um *bin*, o de menor índice terá valor unitário, enquanto os demais, valor nulo. Trata-se de um subterfúgio matemático interessante, já que permite o uso da variável b_i como binária (já que só assume valores 0 e 1), porém deixando-a contínua.

$$0 \leq x_i \leq L - l_i \quad i = 1, \dots, n, \quad (3.28)$$

$$0 \leq y_i \leq W - w_i \quad i = 1, \dots, n, \quad (3.29)$$

As restrições (3.28) e (3.29) garantem que os itens estejam posicionados integralmente no interior do *bin*, ou seja, as coordenadas de seus cantos inferiores-esquerdos devem ser superiores ou iguais a 0 e inferiores ou iguais aos limites do *bin* descontados da dimensão do próprio item. Isto é está retratado em (3.28) para o comprimento (coordenada x) e em (3.29) para a largura (coordenada y).

Por fim, as declarações (3.30) e (3.31) apenas definem as variáveis de decisão utilizadas como reais e binárias, respectivamente.

Para ilustrar melhor o funcionamento do modelo, imaginemos um exemplo didático simples, com um conjunto de dados exagerado. Sejam os dados mostrados na Figura 3.1, descritos a seguir: $W = 100$, $L = 100$, $st = 1$, $w_1 = w_2 = w_3 = w_4 = l_1 = l_2 = l_3 = l_4 = 40$, $d_1 = 5$, $d_2 = 10$, $d_3 = 15$, $d_4 = 20$, $p_1 = p_2 = p_3 = p_4 = 1$.

Figura 3.1 – Exemplo de conjunto de dados para o problema (exceto custos)

Na Figura 3.2 é ilustrado como o modelo programa as ordens no tempo, otimizando a penalidade total de acordo com os diferentes cenários de custos de matéria-prima, atraso e adiantamento.

Figura 3.2 – Resultados de programações de acordo com os custos de matéria-prima, adiantamento e atraso

Interessante observar que, à medida que o custo da matéria-prima aumenta, tende-se a concentrar os itens em um só *bin*, ao passo que com penalidades de atraso e adiantamento maiores, a tendência é que a data de entrega seja perseguida.

4. MÉTODOS HEURÍSTICOS

Apresentada a formulação no capítulo anterior, fica claro que se trata de um problema com grande combinatório, de forma que a otimização será inviabilizada em termos de capacidade computacional de processamento. Embora não haja uma demonstração formal de que é NP-hard, vale ressaltar que trata-se de uma combinação de 2 problemas que são NP-hard, como mostrado por Azizoglu, Kondacki e Kirca (1991) no caso do problema de *scheduling* em ambiente de máquina única com objetivo de minimizar a soma das penalidades por atraso e adiantamento; e Garey e Johnson (1979) no caso do problema de *bin packing* bidimensional orientado sem restrição de corte guilhotinado.

Desta forma, a metodologia do trabalho – complementarmente à proposição da formulação do problema feita no capítulo 3 – envolve a proposição de um método de solução para o problema.

Neste capítulo 4 são apresentados:

- a) **Heurística construtiva:** baseada nas características físicas e lógicas estudadas do problema (bem como dos dois problema associados), tem o objetivo de gerar uma solução viável para o problema com baixo esforço computacional, com a melhor qualidade possível;
- b) **Heurística para obtenção de *lower bound*:** como não é possível obter o ótimo para o problema proposto em tempo computacional razoável, mesmo para instâncias pequenas, a avaliação das soluções obtidas pela heurística construtiva dependem de um bom limitante inferior para o problema. Para isso, propõe-se aqui um método de geração deste *lower bound* de maneira competitiva em relação aos pacotes de otimização de mercado, especificamente para esta formulação.

4.1. Heurística construtiva

A apresentação do método feito inicia-se com a heurística construtiva, que, como dito anteriormente, objetiva a geração de uma solução viável com função objetiva melhorada por ideias adquiridas do estudo do problema, suas características físicas e matemáticas.

Inicialmente, as intuições que pautam o método descrito são colocadas, posteriormente seguidas por algumas demonstrações matemáticas de propriedades de subproblemas envolvidos. Por fim, levando estes subsídios em consideração, o algoritmo da heurística é apresentado.

4.1.1. Intuição e bases teóricas

A solução para o problema cuja formulação é proposta no capítulo 3 desta dissertação envolve, de maneira simplificada, duas grandes decisões:

- Quais itens serão os conjuntos de itens selecionados para formar um lote;
- Dados estes grupos, em qual instante inicia-se seu processamento.

Tomando a Figura 4.1 como uma ilustração de uma série de itens a serem atendidos ao longo do tempo, a primeira intuição a ser adotada é que itens adjacentes tem uma tendência a ter um menor desvio de datas se empacotados em grupos de datas próximas.

Figura 4.1 – Exemplo de instância com os prazos ilustrados graficamente em uma linha do tempo (Fonte: elaboração pelo autor)

A Figura 4.2 seria uma retratação desta intuição, em que seriam feitos encaixes de peças adjacentes em prazo para formar grupos que tendem a melhorar (diminuir) a penalização por desvio ponderado de datas. É trivial mostrar que esta ideia não leva necessariamente à combinação ótima, bastando imaginar uma instância em que um determinado item i tenha penalização por adiantamento nula ($\alpha_i = 0$) e uma alta penalização por atraso ($\beta_i \gg 0$). Neste

caso, este item facilmente poderia ser empacotado em qualquer lote anterior a sua data de entrega, mesmo que esta esteja muito remotamente anterior.

Figura 4.2 – Ilustração de uma ideia intuitiva de empacotamento (Fonte: elaboração do autor)

A união de itens adjacentes em data tende a melhorar a qualidade dos desvios, embora não garanta otimalidade. Por isso, a ideia será utilizada para contemplar, inicialmente, o subitem **a** das proposições de decisão expostas neste item 4.1.1.

Quanto ao subitem **b**, pode-se mostrar que, dado um conjunto de itens definidos que formam um lote, é simples encontrar seu melhor instante de término em tempo polinomial.

Seja um conjunto de n itens i ($i \in \{1, \dots, n\}$) cujo empacotamento em um lote é factível. Sejam d_i suas datas de entrega, tendo neste conjunto $d_1 \leq d_2 \leq d_3 \leq \dots \leq d_n$, sem perda de generalidade. Seja α_i suas penalizações unitárias por adiantamento e β_i suas penalizações unitárias por atraso, utilizando a mesma notação do modelo proposto no capítulo 3. Seja t o instante de término de execução do lote formado pelos n itens e $P(t)$ a penalização por desvio de datas ponderadas por suas penalizações. Seja k ($k \in \{1, \dots, n\}$) o último item com $d_k \leq t$. Temos que:

$$P(t) = \sum_{i=1}^k (t - d_i)\alpha_i + \sum_{i=k+1}^n (d_i - t)\beta_i \quad (4.1)$$

Utilizando $P(t)^*$ como notação para a data de término que minimiza as penalizações por desvio de datas ponderadas por penalizações de um dado conjunto de itens pré-definidos, temos:

Propriedade 4.1: $P(t)^*$ ocorre, necessariamente, com $t = d^*$, sendo $d^* \in \{d_1, d_2, d_3, \dots, d_n\}$, exceto se houver uma restrição externa que impeça que a operação termine nesta data (como, por exemplo, iniciar antes do instante zero).

Prova: suponha que exista t' , tal que $t' > d_k$ e $t' < d_{k+1}$. Aplicando d_k e d_{k+1} na equação 4.1, temos:

$$P(d_k) = \sum_{i=1}^k (d_k - d_i)\alpha_i + \sum_{i=k+1}^n (d_i - d_k)\beta_i \quad (4.2)$$

$$P(d_k) = \sum_{i=1}^k (d_k - t')\alpha_i + \sum_{i=1}^k (t' - d_i)\alpha_i + \sum_{i=k+1}^n (d_i - t')\beta_i + \sum_{i=k+1}^n (t' - d_k)\beta_i \quad (4.3)$$

$$\begin{aligned} P(d_k) &= P(t') + (d_k - t') \sum_{i=1}^k \alpha_i + (t' - d_k) \sum_{i=k+1}^n \beta_i \\ &= P(t') + (t' - d_k) \left(\sum_{i=k+1}^n \beta_i - \sum_{i=1}^k \alpha_i \right) \end{aligned} \quad (4.4)$$

Analogamente, temos que:

$$P(d_{k+1}) = \sum_{i=1}^k (d_{k+1} - d_i)\alpha_i + \sum_{i=k+1}^n (d_i - d_{k+1})\beta_i \quad (4.5)$$

$$\begin{aligned}
P(d_{k+1}) &= \sum_{i=1}^k (d_{k+1} - t')\alpha_i + \sum_{i=1}^k (t' - d_i)\alpha_i + \sum_{i=k+1}^n (d_i - t')\beta_i \\
&\quad + \sum_{i=k+1}^n (t' - d_{k+1})\beta_i
\end{aligned} \tag{4.6}$$

$$\begin{aligned}
P(d_{k+1}) &= P(t') + (d_{k+1} - t') \sum_{i=1}^k \alpha_i + (t' - d_{k+1}) \sum_{i=k+1}^n \beta_i \\
&= P(t') + (d_{k+1} - t') \left(\sum_{i=1}^k \alpha_i - \sum_{i=k+1}^n \beta_i \right)
\end{aligned} \tag{4.7}$$

Como α_i e β_i são não-negativos para qualquer i ($i \in \{1, \dots, n\}$) e $d_{k+1} > t' > d_k$, decorre de das equações (4.4) e (4.7) que, necessariamente $P(d_k) < P(t')$ ou $P(d_{k+1}) < P(t')$. O primeiro caso ocorre quando $\sum_{i=k+1}^n \beta_i < \sum_{i=1}^k \alpha_i$ e o segundo caso ocorre quando $\sum_{i=k+1}^n \beta_i > \sum_{i=1}^k \alpha_i$. $\square$

Claramente, no problema estudado temos uma questão adicional de que o início da operação do lote pode estar restrito por uma limitação de não-sobreposição de tempo, já que uma determinada data pode levar a um início antes do instante 0 ou mesmo sobrepondo o fim de um lote anterior. Entretanto, temos da Propriedade 4.1 que, salvo uma restrição para o início da operação do lote, o melhor instante de término, $P(t)^*$, coincide com algum dos prazos dos itens do conjunto em análise.

Aprofundando esta análise, podemos enunciar outra propriedade:

Propriedade 4.2: Seja um conjunto de n itens i ($i \in \{1, \dots, n\}$), existe um item k tal que $P(k) \leq P(k-1) \leq P(k-2) \leq \dots \leq P(1)$ e, também, $P(k) \leq P(k+1) \leq P(k+2) \leq \dots \leq P(n)$.

Em outras palavras, essa propriedade enuncia que a função $P(t)$, que é uma função linear por partes, tem um ponto de mínimo (k), sendo que, para $i < k$, a função é sempre decrescente no intervalo de 0 a k , conforme i aumenta; ao passo que é sempre crescente, também com o aumento de i . O formato ilustrativo da função $P(t)$ está ilustrado na Figura 4.3.

Figura 4.3 – Ilustração da função $P(t)$, que tem formato linear por partes, concavidade para cima, tendo um ponto de mínimo, antes do qual é sempre decrescente e depois do qual é sempre crescente (Fonte: elaboração do autor)

Prova: Provar a Propriedade (4.2) é equivalente a provar que se $P(d_j) \leq P(d_{j+1})$, então $P(d_{j+1}) \leq P(d_{j+2})$, $\forall j, j \in \{1, \dots, n-2\}$; e, ao mesmo tempo, que se $P(d_j) \leq P(d_{j-1})$, então $P(d_{j-1}) \leq P(d_{j-2})$, $\forall j, j \in \{3, \dots, n\}$.

Utilizando o mesmo raciocínio que levou à equação (4.4) e reforçando que $d_{j+1} \geq d_j$, temos:

$$P(d_{j+1}) = P(d_j) + (d_{j+1} - d_j) \left(\sum_{i=1}^j \alpha_i - \sum_{i=j+1}^n \beta_i \right) \quad (4.8)$$

$$P(d_{j+2}) = P(d_{j+1}) + (d_{j+2} - d_{j+1}) \left(\sum_{i=1}^{j+1} \alpha_i - \sum_{i=j+2}^n \beta_i \right) \quad (4.9)$$

Se $P(d_j) \leq P(d_{j+1})$, então decorre imediatamente de (4.8) que:

$$\sum_{i=1}^j \alpha_i \geq \sum_{i=j+1}^n \beta_i \quad (4.10)$$

Como α_i e β_i são não-negativos para qualquer i ($i \in \{1, \dots, n\}$), em particular, α_{j+1} e β_{j+1} são não-negativos. Desta observação, verificamos que:

$$\sum_{i=1}^j \alpha_i + \alpha_{j+1} \geq \sum_{i=j+1}^n \beta_i - \beta_{j+1} \quad (4.11)$$

E que, portanto:

$$\sum_{i=1}^{j+1} \alpha_i \geq \sum_{i=j+2}^n \beta_i \quad (4.12)$$

Assim, unindo (4.12) a (4.9), temos que se $P(d_j) \leq P(d_{j+1})$, então $P(d_{j+2}) \leq P(d_{j+1})$, $\forall j, j \in \{1, \dots, n-2\}$.

Analogamente, partindo novamente da equação (4.4) e tendo em vista que $d_j \geq d_{j-1}$, temos:

$$P(d_j) = P(d_{j-1}) + (d_j - d_{j-1}) \left(\sum_{i=1}^{j-1} \alpha_i - \sum_{i=j}^n \beta_i \right) \quad (4.13)$$

$$P(d_{j-1}) = P(d_{j-2}) + (d_{j-1} - d_{j-2}) \left(\sum_{i=1}^{j-2} \alpha_i - \sum_{i=j-1}^n \beta_i \right) \quad (4.14)$$

Se $P(d_j) \leq P(d_{j-1})$, então decorre imediatamente de (4.13) que:

$$\sum_{i=1}^{j-1} \alpha_i \leq \sum_{i=j}^n \beta_i \quad (4.15)$$

Como α_i e β_i são não-negativos para qualquer i ($i \in \{1, \dots, n\}$), em particular, α_{j-1} e β_{j-1} são não-negativos. Desta observação, verificamos que:

$$\left(\sum_{i=1}^{j-1} \alpha_i \right) - \alpha_{j-1} \leq \left(\sum_{i=j}^n \beta_i \right) + \beta_{j-1} \quad (4.16)$$

E que, portanto:

$$\sum_{i=1}^{j-2} \alpha_i \leq \sum_{i=j-1}^n \beta_i \quad (4.17)$$

Assim, unindo (4.17) a (4.14), temos que se $P(d_j) \leq P(d_{j-1})$, então $P(d_{j-1}) \leq P(d_{j-2}), \forall j, j \in \{3, \dots, n\}$. $\square$

Sendo assim, das Proposições 4.1 e 4.2, decorre que a decisão enunciada na subitem **b** do início deste item – ou seja, a decisão do instante de início de um conjunto de ordens – é uma consequência do conjunto de itens selecionados para um determinado lote. Pela Proposição 4.1 conclui-se que existe uma data ideal para o término do lote e esta data coincide com a data de entrega de um dos itens deste. Pela Proposição 4.2 é possível concluir não só que existe um mínimo da função $P(t)$, como também que existe um item k , tal que para $t \leq k$, $P(t)$ é decrescente e, para $t \geq k$, $P(t)$ é crescente. Esta propriedade permite encontrar k em tempo computacional polinomial.

Ainda pela Proposição 4.2, podemos facilmente intuir que, caso haja um impedimento de início da operação do lote na data d_k , a data com menor penalização para aquele lote passa a ser a data mais cedo possível, pois a partir de d_k a função de penalização é crescente. Aqui atenta-se para o fato de que isto é verdade considerando a restrição de início como rígida. Obviamente, na busca de melhores soluções, é possível antecipar um lote de peças anterior (penalizando-o) de forma a melhorar a solução do problema.

4.1.2. Algoritmo

Tendo em vista as intuições e conclusões do item 4.1.1, nesta subsecção apresenta-se o algoritmo da heurística construtiva criada e uma ilustração do funcionamento desta.

Seja I um conjunto de itens i ($i \in \{1, \dots, n\}$), caracterizado por suas dimensões, prazo, tempo de processamento, penalidades de adiantamento e atraso e *bin* em que será inserido ($l_i, w_i, d_i, p_i, \alpha_i, \beta_i, bin_i$). Seja P um conjunto de *bins* disponíveis caracterizadas por suas dimensões e um número (l_p, w_p, cod_p), respectivamente o comprimento, largura do *bin* e código do *bin*.

O algoritmo encontra-se no Quadro 4.1.

1. Incluir no conjunto P um *bin* de dimensões L por W ($P \leftarrow (L, W, 1)$)
2. Ordenar os itens de I em ordem crescente de d_i e identifica-los com bin_i crescente
3. Posicionar o item 1 no *bin* disponível em P ($bin_i \leftarrow 1$)
 - 3.1. Inserir dois novos *bins* no conjunto P : ($l_i, W - w_i, 1$) e ($L - l_i, W, 1$)
4. Para cada item i
 - 4.1. Procurar um *bin* no conjunto P com $w_i \leq w_p$ e $l_i \leq l_p$
 - 4.1.1. Se existe um *bin* p , nestas condições, com $w_p \neq W$
 - 4.1.1.1. Fazer $bin_i \leftarrow cod_p$
 - 4.1.1.2. Fazer $l_p \leftarrow l_p - l_i$
 - 4.1.2. Caso contrário, se existe *bin* p com $w_p = W$ e $l_i \leq l_p$
 - 4.1.2.1. Fazer $bin_i \leftarrow cod_p$
 - 4.1.2.2. Inserir um novo *bin* no conjunto P : ($l_i, W - w_i, cod_p$)
 - 4.1.2.3. Atualizar o *bin* p : (l_p, w_p, cod_p) $\leftarrow$ ($l_p - l_i, w_p, cod_p$)
 - 4.1.3. Caso contrário (não haja um *bin* p com $w_i \leq w_p$ e $l_i \leq l_p$)
 - 4.1.3.1. Fazer $bin_i \leftarrow cod_p + 1$
 - 4.1.3.2. Inserir um novo *bin* no conjunto P : ($l_i, W - w_i, bin_i$)
 - 4.1.3.3. Inserir outro novo *bin* no conjunto P : ($L - l_i, W, bin_i$)
 - 4.1.3.4. Retirar de P todos os *bins* com $cod_p \neq bin_i$
 5. Ordenar os itens de I em ordem crescente de bin_i
 6. Para cada subconjunto de itens com mesmo bin_i
 - 6.1. Calcular a data ideal (d^*) de término de acordo com as Propriedades 4.1 e 4.2
 - 6.2. Se a data encontrada for inviável por sobreposição temporal (d^* menor que a data de término de um lote subtraído de seu tempo de processamento) a um lote anterior, iniciar ao término do lote anterior.

Quadro 4.1 – Algoritmo da heurística construtiva proposta

Para melhor ilustrar o algoritmo, a Figura 4.4 ilustra os passos de 1 a 3 do algoritmo, ou seja, sua fase de inicialização.

1. Incluir no conjunto P um *bin* de dimensões L por W ($P \leftarrow (L, W, 1)$)

$$P = \{ \text{bin}(L, W, 1) \}$$

2. Ordenar os itens de I em ordem crescente de d_i

3. Posicionar o item 1 no *bin* disponível em P ($\text{bin}_i = 1$)

- 3.1. Inserir dois novos *bins* no conjunto P: $(l_1, W - w_1, 1)$ e $(L - l_1, W, 1)$

$$P = \{ \text{bin}(l_1, W - w_1, 1), \text{bin}(L - l_1, W, 1) \}$$

Figura 4.4 – Passos 1 a 3 do algoritmo da heurística construtiva

A Figura 4.5 ilustra as 3 possibilidades dos passo 4: (i) ser alocado em um sub-*bin* com largura menor que W ; (ii) ser alocado em um sub-*bin* com largura igual a W ; ou (iii) ser alocado em um novo *bin*. Interessante observar que existe uma prioridade para ocupar retalhos gerados por um determinado item anteriormente; posteriormente a utilizar o restante do *bin*. Só quando nenhum sub-*bin* do conjunto P abarcar o item i é que recorre-se a uma nova placa. Neste caso, todos os sub-*bins* disponíveis são esquecidos.

Figura 4.5 – Ilustração do passo 4 do algoritmo e suas três ramificações
(Fonte: elaboração do autor)

Por fim, nos passos 5 e 6, faz-se uma análise dos itens que foram alocados em um mesmo *bin* e faz-se sua alocação temporal. A Figura 4.6 ilustra o processo de escolha da melhor data utilizando as Propriedades enunciadas anteriormente (embora computacionalmente esta escolha não seja feita através de enumeração explícita das penalidades nos prazos dos itens, mas sim através de um método de busca que explora a concavidade da função penalização).

Importante ressaltar o item 6.2 do algoritmo, que não trata da sobreposição temporal investigando pós-otimizações, apenas desloca a data de término do mínimo possível para evitar a sobreposição temporal (já que a função é crescente a partir do ponto encontrado).

Figura 4.6 – Ilustração do passo 6 do algoritmo
(Fonte: elaboração do autor)

4.2. Heurística para obtenção do *lower bound*

O intuito da geração de um limitante inferior para as instâncias do problema é duplo:

- Avaliar a heurística construtiva proposta na subseção 4.1;
- Avançar no conhecimento do problema, melhorando limitantes gerados por pacotes de otimização de mercado através das propriedades demonstradas anteriormente.

O método se baseia na ideia de inserir um item de cada vez, gerando soluções parciais cujas penalidades só podem aumentar. Isto é, o item é posicionado em todas as suas combinações possíveis em relação aos itens anteriormente alocados e o instante de término resultante é aquele determinado pelas Propriedades 4.1 e 4.2. Assim, quando um novo item é adicionado,

a penalidade de uma solução parcial só pode aumentar (seja por acrescentar um novo lote, seja por desviar o atendimento de datas dos itens ali contidos). Esse método é válido desde que, a cada inserção de um novo item, este gere novas soluções parciais com todos os lotes existentes ou em um novo lote sozinho, explorando todas as ramificações do combinatório do problema.

Assim, aproxima-se de um método exato na questão da combinação dos itens em lotes, porém não pode ser caracterizado como tal porque não explora todas as possibilidades de alocação de datas dos lotes formados. Entretanto, como trata-se de gerar um limitante inferior, na medida em que são permitidas sobreposições temporais entre lotes, utilizando-se das Propriedades 4.1 e 4.2 para encontrar a data ideal (com a restrição de que não seja iniciada antes do instante 0, pois neste caso a nova data ideal é trivial).

Seja I o conjunto dos i itens a serem programados e empacotados, tendo como dados a sua data de execução d_i , sua penalidade unitária por adiantamento α_i , sua penalidade unitária por atraso β_i e seu tempo de processamento p_i . Seja S o conjunto de soluções parciais geradas pela heurística, contendo as penalidades pela alocação de alguns dos itens alocados nos lotes $P(s)$. Considerando um custo unitário por *bin* de θ e um *setup* de execução por lote st , o algoritmo de geração do limitante inferior para o problema proposto encontra-se no Quadro 4.2.

1. Gerar solução factível (UB_{inst})
2. Gerar limitante inferior do problema de *bin packing* associado à instância, desconsiderando datas (LB_{bin})
3. Ordenar os itens de I em ordem crescente de d_i
4. Criar uma solução parcial em S
 - 4.1. Criar um único lote nesta solução, alocando o item 1 a ele
 - 4.2. Programar esse lote com início em $t_0 = \max(0, d_1 - st - p_1)$
 - 4.3. Atualizar penalidade por uso de *bins* da solução 1: $P_{bin,1} = \theta$
 - 4.4. Atualizar penalidade por desvio de datas da solução 1:

$$P_{data,1} = \alpha_1 \max(0, d_1 - t_0) + \beta_1 \max(0, t_0 - d_1)$$
5. Para cada item $i \in I$
 - 5.1. Para cada solução parcial $s \in S$
 - 5.1.1. Gerar uma nova solução parcial s' idêntica a s com i em um novo lote
 - 5.1.1.1. Programar o novo lote com início em $t_0 = \max(0, d_i - st - p_i)$
 - 5.1.1.2. Atualizar penalidade por uso de *bins* de s' : $P_{bin,s'} = P_{bin,s} + \theta$
 - 5.1.1.3. Atualizar penalidade por desvio de datas da solução s' :

$$P_{data,s'} = P_{data,s} + \alpha_i \max(0, d_i - t_0) + \beta_i \max(0, t_0 - d_i)$$
 - 5.1.1.4. Inserir s' em S se $UB_{inst} \leq \max(P_{bin,s'}, \theta \times LB_{bin}) + P_{data,s'}$
 - 5.1.2. Para cada lote p em $P(s)$
 - 5.1.2.1. Se i pode ser empacotada no lote p na solução s , gerar uma nova solução parcial s' com os mesmos lotes e itens de s , acrescentando o item i no lote p
 - 5.1.2.1.1. Calcula nova data ideal de início do lote alterado (t_0) em substituição à antiga data ideal antes da entrada do item i (t_A)
 - 5.1.2.1.2. Fazer penalidade por uso de *bins* da solução s' igual à de s , pois não há acréscimo de novo lote ($P_{bin,s'} = P_{bin,s}$)
 - 5.1.2.1.3. Fazer penalidade por desvio de datas da solução s' igual à de s , acrescentando a penalidade pelo item i

$$(P_{data,s'} = P_{data,s} + \alpha_i \max(0, d_i - t_0) + \beta_i \max(0, t_0 - d_i))$$
 - 5.1.2.1.4. Atualizar penalidade dos itens ip que estão no lote p :

$$(P_{data,s'} = P_{data,s} + \alpha_{ip} \max(0, d_{ip} - t_0) + \beta_{ip} \max(0, t_0 - d_{ip}) - \alpha_{ip} \max(0, d_{ip} - t_A) + \beta_{ip} \max(0, t_A - d_{ip}))$$
 - 5.1.2.1.5. Inserir s' em S se $UB_{inst} \leq \max(P_{bin,s'}, \theta \times LB_{bin}) + P_{data,s'}$
 6. Encontrar a menor soma de penalidades com *bins* e desvios de datas dentre as soluções parciais em S ($LB_{inst} = \min_{s \in S} (P_{data,s} + \max(\theta \times LB_{bin}, P_{bin,s}))$)

Quadro 4.2 – Algoritmo da heurística de geração do *lower bound* do problema

Alguns dos procedimentos que envolvem parametrização e explicação dos métodos utilizados são detalhados a seguir:

- No passo 1 do algoritmo é gerado um limitante superior para o problema (uma solução factível de qualidade interessante) de forma a possibilitar, em passos posteriores, a eliminação de soluções parciais que tenham penalidade superior à penalidade deste *upper bound*. Neste caso, utilizou-se a heurística construtiva apresentada na subseção 4.1;

- Para cada instância, o conjunto de itens proposto está associado a um problema de empacotamento (*bin packing*) em unidades de tamanho (L, W). Assim, no passo 2 do algoritmo, calcula-se um limitante inferior para este problema associado, de forma que, se a penalidade com uso de *bins* de uma solução parcial for inferior a este limitante, este é considerado como a penalidade por uso de *bins* para aquela solução parcial, melhorando sua qualidade como limitante inferior para este problema. Neste caso, utilizou-se como limitante inferior o maior valor entre o *lower bound* contínuo (área dos itens dividida pela área de um *bin*, arredondando o resultado para o inteiro imediatamente superior) e a contagem de itens maiores do que a metade do *bin* em ambas dimensões;
- O passo 4 inicia o conjunto de soluções parciais com o item de prazo mais curto posicionado para terminar em sua data de entrega (se isto for impossível, por necessitar de um início antes do instante zero, será finalizada com o mínimo de atraso possível);
- Finalmente, o passo 5 é um procedimento a ser feito para cada item que passa a figurar nas soluções parciais: este é combinado em todas as possibilidades com os itens que já se encontram tratados. Para cada uma das soluções já existentes no conjunto S, existem duas possibilidades distintas a serem tratadas, gerando novas soluções parciais decorrentes desta:
 - O item passa a formar um novo lote sozinho (passo 5.1). Neste caso não há necessidade de verificar a viabilidade de encaixe e a penalidade adicional é o custo do *bin* (θ), além de um eventual deslocamento de sua data de término por necessidades começar antes do instante zero para atendê-la; ou
 - O item será inserido em cada um dos *bins* existentes na solução parcial analisada (passo 5.1.2.1), gerando uma nova para cada combinação desta. Aqui vale destacar alguns procedimentos:
 - No passo 5.1.2.1 é analisada a viabilidade de entrada do item em um determinado lote. Aqui, esta verificação é feita apenas pela viabilidade de área, já que tratamos de um limitante inferior;
 - A data ideal de finalização do lote é calculada utilizando as Propriedades 4.1 e 4.2 aplicadas aos itens do *bin* adicionados do item a ser inserido. A data de início (passo 5.1.2.1.1) é uma decorrência desta,

descontando os tempos de processo dos itens envolvidos e o *setup* (st) do *bin*.

Desta forma, temos um limitante inferior para o problema apresentado para qualquer número de itens que seja inserido. Isso é válido porque, até a inserção de um determinado item, todas as combinações de itens estão sendo abarcadas, posicionando-os em suas datas de menor penalidade. A entrada de um item adicional em um certo lote só pode piorar ou manter a penalidade. Como se trata de um *lower bound* não foi tratada de maneira heurística a sobreposição temporal dos lotes.

A Figura 4.7 ilustra o processo da heurística para 3 itens.

Figura 4.7 – Ilustração da ramificação de exploração do processo de geração do *lower bound*
(Fonte: elaboração do autor)

5. EXPERIMENTOS COMPUTACIONAIS

O quinto capítulo deste trabalho relata o procedimento experimental realizado. Foram realizados experimentos computacionais com uma massa de dados desenvolvida para o problema, já que se trata de uma abordagem pioneira na literatura. Inicialmente, é descrito o processo de criação destas instâncias, baseadas em instâncias já existentes na literatura. Posteriormente, é descrita a forma como estes experimentos foram realizados e avaliados. Por fim, apresentam-se os resultados obtidos.

5.1. Geração de instâncias

Como reforçado nos capítulos iniciais deste trabalho, a formulação aqui tratada é uma abordagem pioneira, embora tangencie dois problemas clássicos na literatura.

De fato, não existe um conjunto de instâncias prontas para testar as heurísticas construtivas criadas. Assim, como parte deste trabalho, foi criada uma massa de dados própria para o problema, intencionando não só realizar tal teste, como também propor um conjunto de instâncias que pode vir a ser utilizado em futuras pesquisas, tendo já um conjunto de soluções a serem confrontadas.

Por outro lado, optou-se aqui por aproveitar dados de instâncias renomadas da literatura de *bin packing*, que são referência para teste de algoritmos, heurísticas e metaheurísticas na área. Trata-se de um conjunto teste bastante conhecido e utilizado na literatura, que retrata diversos cenários de tamanhos de itens.

Este conjunto apresentado por Lodi, Martello e Vigo (1999) é composto por 500 instâncias para o problema de *bin packing*, sendo 10 classes de problemas em termos de tamanho dos itens, com 5 portes (em termos de número de itens em cada instância) e 10 repetições. Devido à necessidade de rodar diversos experimentos e o esforço computacional demandado por cada um deles, optou-se aqui por um recorte, selecionando 4 classes de problemas com 4 repetições de cada em 5 portes.

As dimensões dos *bins* e dos itens em cada uma das classes propostas são geradas da seguinte forma (LODI, MARTELLO e VIGO, 1999). Importante mencionar que manteve-se aqui a numeração das classes do artigo original, daí o seu início na classe 7:

- Classe 7 (itens “largos”): $L = W = 100$; l_i gerados aleatoriamente seguindo uma distribuição $U[1L, 1/2L]$; w_i gerados aleatoriamente seguindo uma distribuição $U[2/3W, W]$;
- Classe 8 (itens “estreitos”): $L = W = 100$; l_i gerados aleatoriamente seguindo uma distribuição $U[2/3L, L]$; w_i gerados aleatoriamente seguindo uma distribuição $U[1, 1/2W]$;
- Classe 9 (itens “grandes”): $L = W = 100$; l_i gerados aleatoriamente seguindo uma distribuição $U[1/2L, L]$; w_i gerados aleatoriamente seguindo uma distribuição $U[1/2W, W]$;
- Classe 10 (itens “pequenos”): $L = W = 100$; l_i gerados aleatoriamente seguindo uma distribuição $U[1, 1/2L]$; w_i gerados aleatoriamente seguindo uma distribuição $U[1, 1/2W]$.

Para cada uma destas classes foram selecionadas 4 repetições de cada um dos 3 portes selecionados entre os propostos por estes autores: 20 itens, 60 itens e 100 itens. Os valores gerados nos artigos em que foram apresentados encontram-se disponíveis na página do grupo de Pesquisa Operacional do departamento de Eletrônica e Informática da Universidade de Bologna: <http://www.or.deis.unibo.it/research.html>.

Entretanto, além destes dados, as instâncias para o problema de programação de operações de corte deste trabalho necessitam de outros dados a respeito dos *bins* e das ordens. Estes foram gerados da seguinte maneira:

- Os tempos de processamento dos itens (p_i) para todas as instâncias foram gerados aleatoriamente seguindo uma distribuição $U[1, 20]$;
- Para as datas de entrega (d_i) foram propostos 3 cenários:
 - **Prazos estreitos:** gerados aleatoriamente seguindo uma distribuição $U[5\%\sum_i p_i, 25\%\sum_i p_i]$;
 - **Prazos dilatados:** gerados aleatoriamente seguindo uma distribuição $U[65\%\sum_i p_i, 85\%\sum_i p_i]$;
 - **Prazos variados:** gerados aleatoriamente seguindo uma distribuição $U[15\%\sum_i p_i, 85\%\sum_i p_i]$.

- Para os tempos de *setup* dos *bins*, são propostos 2 cenários baseados no tempo de operação médio dos itens da instância:
 - **Setup curto:** $\bar{p}$;
 - **Setup longo:** $10 \bar{p}$.
- Para as proporções entre os custos de atraso, atendimento e uso dos *bins*, foram propostos 4 cenários:
 - **Custos das penalidades de atraso e de adiantamento baixos em relação à matéria-prima:** $\theta = 10.000$; α_i gerados aleatoriamente seguindo uma distribuição $U[1, 5]$; β_i gerados aleatoriamente seguindo uma distribuição $U[1, 5]$;
 - **Custos das penalidades de atraso altos e de adiantamento baixos em relação à matéria-prima:** $\theta = 10.000$; α_i gerados aleatoriamente seguindo uma distribuição $U[1, 5]$; β_i gerados aleatoriamente seguindo uma distribuição $U[25, 75]$;
 - **Custos das penalidades de atraso baixos e de adiantamento altos em relação à matéria-prima:** $\theta = 10.000$; α_i gerados aleatoriamente seguindo uma distribuição $U[25, 75]$; β_i gerados aleatoriamente seguindo uma distribuição $U[1, 5]$;
 - **Custos das penalidades de atraso e de adiantamento altos em relação à matéria-prima:** $\theta = 10.000$; α_i gerados aleatoriamente seguindo uma distribuição $U[25, 75]$; β_i gerados aleatoriamente seguindo uma distribuição $U[25, 75]$;

Assim, para cada uma das 4 classes, foram criados 24 cenários diferentes em relação a custos das penalidades, tempos de *setup* e folga nos prazos. Uma vez que existem 3 portes distintos com 4 repetições de cada classe, chega-se a um conjunto de instâncias de 1.152 experimentos. O código fonte para a geração das instâncias encontra-se no apêndice A.

O intuito da extensão e variedade da massa de dados é permitir explorar diferentes cenários, não só para verificar o comportamento dos gaps em cada um deles, como também a robustez dos métodos de solução propostos.

5.2. Procedimento experimental

Como se trata de um problema NP-difícil de alta complexidade, buscou-se comparar o desempenho dos métodos propostos em relação ao obtido por um *software* de mercado de alto desempenho.

Para tanto, as instâncias geradas foram rodadas no CPLEX 12.3, utilizando como interface o IBM ILOG CPLEX Optimization Studio. As rodadas foram feitas em um computador cujo processador é i7 2ª geração, com memória RAM 8 Gb. Tais instâncias foram submetidas ao *software* com a formulação proposta no capítulo 3 durante 10 minutos, sendo coletados seus limitantes superiores (UB), inferiores (LB), assim como as penalidades por atraso, adiantamento e utilização de *bins* na solução. O código fonte desta implementação encontra-se no apêndice B (estando ali o script utilizado, o modelo implementado e um exemplo de arquivo de dados de instância).

Por outro lado, utilizando o mesmo computador para manter a equidade de condições, as heurísticas propostas no capítulo 4 foram implementadas em *Visual Basic for Applications* (VBA) com interface no Excel 2010. Neste caso, o limitante superior foi rodado praticamente instantaneamente, já que é uma heurística construtiva, enquanto o limitante inferior, por ser um método que gera um combinatório muito grande, foi limitado a 60 segundos. O código fonte da heurística construtiva encontra-se no apêndice C, enquanto o código da geração do *lower bound* encontra-se no apêndice D.

De posse de ambas rodadas é possível calcular algumas medidas de desempenho, tanto dos *gaps* gerados, quanto do comparativos entre as soluções propostas. Para medir a qualidade da solução obtida utiliza-se a medida:

$$gap = \frac{UB - LB}{LB} \quad (5.1)$$

Além disto, tanto para os limitantes superior (UB) e inferior (LB), quanto para o próprio gap, foi feita uma comparação entre o resultado obtido pelo otimizador de mercado (CPLEX) e o método proposto. Esta diferença foi medida analogamente pela relação:

$$Diferença_{\%} = \frac{Método\ proposto - Software\ de\ mercado}{Software\ de\ mercado} \quad (5.2)$$

Finalmente, os resultados foram obtidos para todas as instâncias e agrupados de acordo com o tipo de recorte que representam. Também foi computada a quantidade de vezes que o método proposto foi superior ao *software* de mercado para cada tipo de instância.

5.3. Resultados

Levando em conta a massa de dados descrita em 5.1 e o procedimento experimental apresentado em 5.2, são apresentadas aqui tabelas que sumarizam os resultados obtidos. Em negrito estão os resultados superiores na comparação entre o *software* de mercado e os métodos propostos.

Tabela 5-1 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “pequenos” (classe 10)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	57783	21977	212%	87830	51893	70%
		α alto e β baixo	59864	22712	214%	89605	44923	107%
		α baixo e β alto	178425	64583	176%	247381	66720	272%
		α e β altos	178454	57521	214%	224061	66543	238%
	Longo	α e β baixos	67595	31650	128%	115842	53618	118%
		α alto e β baixo	69367	30504	146%	104945	56568	85%
		α baixo e β alto	384283	239744	62%	666329	101062	571%
		α e β altos	298621	189593	59%	539684	90178	500%
	Variado	α e β baixos	58759	23341	157%	72078	47592	55%
		α alto e β baixo	64378	21474	253%	85787	51451	71%
		α baixo e β alto	116859	36592	220%	153370	63076	144%
		α e β altos	122952	37839	227%	159295	63357	153%
Longo	Curto	α e β baixos	63870	27427	160%	106892	53240	102%
		α alto e β baixo	66648	27765	167%	97992	55969	74%
		α baixo e β alto	326632	169612	93%	559741	86991	542%
		α e β altos	305979	162469	88%	541317	88741	510%
	Longo	α e β baixos	57103	21257	223%	81522	45074	85%
		α alto e β baixo	61497	21394	244%	80562	48206	68%
		α baixo e β alto	125742	39369	220%	176204	63794	177%
		α e β altos	132484	43686	207%	186648	63989	192%
	Variado	α e β baixos	71032	31772	132%	107047	59258	80%
		α alto e β baixo	66175	30836	133%	112458	53216	115%
		α baixo e β alto	299515	186483	62%	597908	91105	552%
		α e β altos	302148	169866	78%	538227	89230	498%

Tabela 5-2 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “estreitos” (classe 8)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	78648	17558	368%	118837	65011	84%
		α alto e β baixo	79084	17564	385%	125579	64991	95%
		α baixo e β alto	193841	60776	217%	276370	69221	300%
		α e β altos	194216	60148	226%	264196	69787	278%
	Longo	α e β baixos	92842	27938	241%	161044	66789	141%
		α alto e β baixo	91447	28046	233%	164981	66852	147%
		α baixo e β alto	500764	259858	93%	925836	102548	801%
		α e β altos	463796	226498	106%	857902	98334	772%
	Variado	α e β baixos	75896	18417	346%	109949	64780	70%
		α alto e β baixo	73854	16383	380%	113256	62427	82%
		α baixo e β alto	128741	35794	261%	197333	65810	201%
		α e β altos	132769	34769	283%	189241	65588	190%
Longo	Curto	α e β baixos	93089	27519	240%	153209	66294	132%
		α alto e β baixo	97795	24733	310%	146723	66201	122%
		α baixo e β alto	386756	156737	147%	638109	90278	608%
		α e β altos	403049	159261	155%	720170	88554	715%
	Longo	α e β baixos	78425	16454	419%	123608	64836	91%
		α alto e β baixo	75068	14390	454%	112034	64843	73%
		α baixo e β alto	145508	41839	249%	225668	67178	238%
		α e β altos	145312	38878	277%	222733	66492	235%
	Variado	α e β baixos	89853	25642	263%	160916	66386	144%
		α alto e β baixo	87749	25083	258%	150026	66555	126%
		α baixo e β alto	414827	178471	133%	754787	93734	706%
		α e β altos	452567	201846	125%	858965	97684	778%

Tabela 5-3 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “largos” (classe 7)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	Gap
Curto	Curto	α e β baixos	68728	19911	279%	98586	57416	74%
		α alto e β baixo	70917	19682	290%	98377	62404	58%
		α baixo e β alto	183284	57578	219%	250176	70137	260%
	Longo	α e β altos	166816	55487	201%	238513	69609	244%
		α e β baixos	76995	27710	190%	121905	61110	100%
		α alto e β baixo	84069	29736	190%	131115	63465	108%
		α baixo e β alto	415926	218140	90%	641893	94098	582%
		α e β altos	403534	211762	92%	638812	94219	575%
		α e β baixos	69314	18607	317%	97693	59639	64%
		α alto e β baixo	72236	18837	315%	97766	62155	58%
		α baixo e β alto	120577	33446	270%	165142	66472	149%
	Longo	α e β altos	144718	38904	274%	188532	66668	183%
		α e β baixos	76000	26934	197%	115305	62972	84%
		α alto e β baixo	85951	26738	235%	122894	64395	91%
Variado	Curto	α baixo e β alto	428551	176490	142%	633132	95796	559%
		α e β altos	349024	158900	121%	561984	91746	514%
		α e β baixos	70376	19145	312%	102527	58461	76%
	Longo	α alto e β baixo	71825	18972	312%	95599	59053	63%
		α baixo e β alto	148742	39567	282%	196901	67795	191%
		α e β altos	136381	40282	246%	195589	67568	190%
		α e β baixos	83700	26746	222%	126085	62842	102%
		α alto e β baixo	80843	29169	183%	123745	66654	86%
		α baixo e β alto	415845	187027	125%	638531	93734	582%
		α e β altos	404901	188884	114%	637155	94426	576%

Tabela 5-4 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 20 itens e itens “grandes” (classe 9)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	161942	112106	69%	180084	147894	23%
		α alto e β baixo	174591	104743	113%	179442	147829	23%
		α baixo e β alto	287183	145321	114%	367058	153822	142%
	Longo	α e β altos	287359	148601	106%	351761	154693	132%
		α e β baixos	197503	122304	83%	234898	150535	58%
		α alto e β baixo	197154	119644	85%	230007	150593	55%
	Longo	α baixo e β alto	810045	337833	139%	1159913	184959	544%
		α e β altos	801335	325151	146%	1109727	186385	503%
	Curto	α e β baixos	158653	108832	81%	171963	147550	18%
		α alto e β baixo	159964	116538	61%	177971	150067	21%
		α baixo e β alto	231913	117175	123%	292456	153499	93%
	Longo	α e β altos	235779	128267	105%	296922	151002	101%
Variado	Curto	α e β baixos	190198	113697	92%	219629	154488	45%
		α alto e β baixo	193140	112763	101%	224981	152202	51%
		α baixo e β alto	727814	267347	171%	1073743	177162	512%
	Longo	α e β altos	739931	272688	172%	1041607	177296	495%
		α e β baixos	159861	108988	73%	175493	147614	20%
		α alto e β baixo	161157	112192	69%	181679	147658	25%
		α baixo e β alto	243186	124121	117%	314319	149197	113%
	Curto	α e β altos	246012	131633	106%	302089	149615	105%
		α e β baixos	187572	113286	92%	221124	149812	50%
		α alto e β baixo	194001	116879	92%	232243	149705	57%
	Longo	α baixo e β alto	816167	313669	163%	1183631	190056	531%
		α e β altos	779228	292039	167%	1060467	186918	470%

Tabela 5-5 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “pequenos” (classe 10)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	705629	15105	4896%	281026	96314	193%
		α alto e β baixo	764082	17412	5154%	289978	96293	202%
		α baixo e β alto	2311496	54056	4282%	1349476	100558	1268%
	Longo	α e β altos	2483242	59061	4158%	1422653	100322	1329%
		α e β baixos	1238207	40090	3057%	489287	98088	396%
		α alto e β baixo	1232488	40078	3025%	466084	97705	379%
	Longo	α baixo e β alto	10098329	396381	2448%	4334000	126959	3312%
		α e β altos	10632295	414257	2468%	4650535	129573	3535%
	Longo	α e β baixos	705337	15414	4997%	245697	96088	158%
		α alto e β baixo	795858	15692	5952%	233618	96122	145%
		α baixo e β alto	1718454	25179	7127%	798992	96595	749%
	Longo	α e β altos	2127471	23051	9532%	800261	96871	751%
Variado	Curto	α e β baixos	1169466	21923	5858%	409674	97446	328%
		α alto e β baixo	1230630	24123	5535%	417455	97440	332%
		α baixo e β alto	9277550	120883	7602%	3493181	118381	2842%
	Longo	α e β altos	10257159	133608	7636%	3844217	122524	3038%
		α e β baixos	679838	18439	4322%	257512	96194	169%
		α alto e β baixo	765988	15940	5025%	267212	96156	179%
	Longo	α baixo e β alto	1762381	32008	5896%	1001613	98110	938%
		α e β altos	2220983	31193	7202%	1039319	97603	977%
	Longo	α e β baixos	1154050	26765	4363%	424946	97917	336%
		α alto e β baixo	1366071	28456	5160%	439489	97677	353%
		α baixo e β alto	9332403	190410	4830%	3574298	120922	2861%
	Longo	α e β altos	10627607	213559	4883%	3967690	123851	3094%

Tabela 5-6 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “estritos” (classe 8)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	719621	12758	5541%	423209	138918	207%
		α alto e β baixo	795556	12506	6264%	420591	138863	203%
		α baixo e β alto	2143234	48456	4333%	1585910	143128	1012%
	Longo	α e β altos	2247082	51385	4328%	1555266	142614	990%
		α e β baixos	1194686	33167	3504%	668058	140239	376%
		α alto e β baixo	1212514	33264	3546%	688141	140410	394%
	Longo	α baixo e β alto	10680547	406642	2540%	6040301	168775	3485%
		α e β altos	9787165	382306	2462%	5755879	169442	3302%
	Longo	α e β baixos	661252	10497	6196%	348356	138653	152%
		α alto e β baixo	817972	10451	7737%	358198	138628	160%
		α baixo e β alto	1704639	16880	9992%	1068764	139831	669%
	Longo	α e β altos	1772187	19551	8966%	1049614	139764	654%
Variado	Curto	α e β baixos	1036917	17013	5992%	585727	139976	319%
		α alto e β baixo	1294205	16763	7581%	612393	139926	336%
		α baixo e β alto	8250768	121925	6656%	5208145	164342	3070%
	Longo	α e β altos	9925873	127306	7689%	5542746	164971	3264%
		α e β baixos	646779	11049	5752%	379678	138740	175%
		α alto e β baixo	760631	11073	6763%	376463	138728	172%
	Longo	α baixo e β alto	1810610	22307	8067%	1253122	140244	799%
		α e β altos	2073083	27667	7614%	1357215	140896	869%
	Longo	α e β baixos	1191539	21835	5357%	623769	140092	348%
		α alto e β baixo	1252714	21715	5674%	601566	140469	329%
		α baixo e β alto	9585516	200827	4706%	5430036	167749	3168%
	Longo	α e β altos	10961591	226160	4845%	5892458	170221	3388%

Tabela 5-7 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “largos” (classe 7)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	650356	17347	4041%	331195	135073	146%
		α alto e β baixo	791261	19857	4162%	325327	135088	142%
		α baixo e β alto	2391767	54135	4354%	1517633	139972	987%
	Longo	α e β altos	2357937	56595	4068%	1466301	139205	957%
		α e β baixos	1202589	36416	3253%	576249	136841	322%
		α alto e β baixo	1182151	37287	3116%	557132	136617	310%
	Longo	α baixo e β alto	10131425	392318	2481%	5219789	164544	3076%
		α e β altos	10862555	409352	2554%	5228786	165245	3071%
	Curto	α e β baixos	626682	15635	4324%	304172	134914	126%
		α alto e β baixo	640507	15478	4503%	293485	134885	118%
		α baixo e β alto	1625201	26496	6166%	915281	135882	577%
	Longo	α e β altos	1861829	25796	7210%	944372	135627	600%
Variado	Longo	α e β baixos	1172733	22730	5215%	539614	136574	295%
		α alto e β baixo	1423872	26814	5206%	522403	136369	284%
		α baixo e β alto	9677869	144330	6628%	4200975	162141	2489%
	Curto	α e β altos	10501013	134069	7768%	4674269	157849	2872%
		α e β baixos	658592	20830	3061%	311803	134956	131%
		α alto e β baixo	620215	15969	4172%	308317	134981	129%
	Longo	α baixo e β alto	1801005	30102	6058%	1095057	136650	704%
		α e β altos	1956253	30920	6444%	1169280	137157	758%
		α e β baixos	1229018	31507	3807%	563459	136501	314%
	Curto	α alto e β baixo	1407255	24619	5806%	530207	136602	289%
		α baixo e β alto	10094862	210920	4697%	4769802	164236	2812%
		α e β altos	10011789	200677	4925%	4660626	161787	2786%

Tabela 5-8 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 60 itens e itens “grandes” (classe 9)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	571724	100957	536%	604694	432892	40%
		α alto e β baixo	593650	105924	534%	622560	432826	44%
		α baixo e β alto	2013301	145724	1357%	2002285	438509	358%
	Longo	α e β altos	2097848	153791	1308%	2101562	439500	378%
		α e β baixos	1237466	121668	990%	1086888	435480	150%
		α alto e β baixo	1204840	117528	1050%	1057854	435982	143%
	Longo	α baixo e β alto	10079670	470458	2056%	9033210	481247	1785%
		α e β altos	11258177	505366	2129%	9898841	481916	1956%
	Curto	α e β baixos	561561	103864	546%	577795	432555	34%
		α alto e β baixo	538604	99025	552%	584364	432539	35%
		α baixo e β alto	1623837	107992	1571%	1522671	433474	252%
	Longo	α e β altos	1593287	125742	1297%	1536440	433523	255%
Variado	Longo	α e β baixos	1053551	110813	962%	1032265	434587	138%
		α alto e β baixo	942319	110506	831%	991038	434431	128%
		α baixo e β alto	8467122	226649	3746%	8294781	463321	1692%
	Curto	α e β altos	9430796	288042	3258%	9285292	470809	1874%
		α e β baixos	569678	104929	530%	608454	432656	41%
		α alto e β baixo	571557	97635	572%	596208	432603	38%
		α baixo e β alto	1836060	115290	1694%	1750794	434802	303%
	Longo	α e β altos	1767481	133305	1321%	1740213	434631	300%
		α e β baixos	1127175	119395	901%	1020963	434679	135%
		α alto e β baixo	1077716	116065	958%	1075532	435224	148%
	Longo	α baixo e β alto	8660817	302091	2797%	8341694	467774	1683%
		α e β altos	9847708	342929	2804%	9296463	478353	1845%

Tabela 5-9 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “pequenos” (classe 10)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	1285484	14679	9231%	534086	148599	260%
		α alto e β baixo	1483698	14784	10586%	534990	148596	262%
		α baixo e β alto	5874337	52800	11140%	3536698	153230	2211%
	Longo	α e β altos	6122450	55850	11065%	3548877	153486	2223%
		α e β baixos	2658165	44210	5969%	997660	150307	564%
		α alto e β baixo	2617732	41597	6242%	988829	150221	559%
	Longo	α baixo e β alto	27786542	505838	5387%	11162456	180777	6093%
		α e β altos	28111080	522097	5306%	11372041	176762	6353%
	Longo	α e β baixos	1286743	12904	10754%	418559	148383	183%
		α alto e β baixo	3102881	12984	25766%	426510	148401	189%
		α baixo e β alto	4399307	20351	22206%	1924294	149295	1193%
	Longo	α e β altos	5979042	19619	31938%	1864211	149370	1149%
Variado	Curto	α e β baixos	2586487	18558	14340%	899564	149636	500%
		α alto e β baixo	3415231	19199	18765%	857606	149815	475%
		α baixo e β alto	26994355	124994	21553%	9623733	175878	5374%
	Longo	α e β altos	27987800	121994	23166%	10156252	172296	5808%
		α e β baixos	1282948	13164	10451%	476626	148461	222%
		α alto e β baixo	2414175	13399	19416%	482196	148462	225%
	Longo	α baixo e β alto	4850092	28001	17666%	2695254	150940	1693%
		α e β altos	6183926	25969	24853%	2594147	151361	1627%
	Longo	α e β baixos	2688860	24324	11297%	943593	150112	529%
		α alto e β baixo	3430749	25446	13837%	893692	150168	498%
		α baixo e β alto	27034329	219307	12225%	10751900	175082	6060%
	Longo	α e β altos	25067267	230216	10815%	10023725	176142	5615%

Tabela 5-10 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “estreitos” (classe 8)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	1317142	17323	8922%	758133	229372	231%
		α alto e β baixo	1543191	17321	10443%	694470	229333	203%
		α baixo e β alto	5709091	54235	10772%	4035338	233672	1636%
	Longo	α e β altos	5771092	53711	10861%	3902773	233340	1586%
		α e β baixos	2563290	43798	5913%	1410625	230852	513%
		α alto e β baixo	2748590	46133	6003%	1468129	230880	535%
	Longo	α baixo e β alto	28296220	538546	5159%	15383505	264793	5708%
		α e β altos	29342578	567781	5080%	16115922	262188	6034%
	Curto	α e β baixos	1247894	15408	9831%	604782	229124	164%
		α alto e β baixo	3015610	15432	24102%	645319	229135	182%
		α baixo e β alto	4425197	22388	21435%	2596694	229946	1035%
	Longo	α e β altos	5658041	21944	28256%	2711320	230143	1083%
Variado	Longo	α e β baixos	2589162	22075	12853%	1361232	230674	489%
		α alto e β baixo	3666262	23215	17806%	1430369	230776	520%
		α baixo e β alto	27958535	133475	21399%	15250097	254828	5893%
	Curto	α e β altos	26332399	130000	20225%	14562805	256388	5573%
		α e β baixos	1314511	15793	10095%	667824	229206	191%
		α alto e β baixo	1823654	15822	14819%	678157	229177	196%
	Longo	α baixo e β alto	4536448	28419	17272%	2888747	231269	1148%
		α e β altos	5585556	27727	21173%	3182026	230886	1286%
	Curto	α e β baixos	2539863	25964	10551%	1401089	230603	510%
		α alto e β baixo	3454269	26847	13826%	1356521	230811	491%
		α baixo e β alto	26578469	215201	12398%	14449922	258179	5484%
	Longo	α e β altos	26691316	213683	12467%	13944368	259865	5275%

Tabela 5-11 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “largos” (classe 7)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	1293309	17276	8113%	619908	225633	175%
		α alto e β baixo	1483405	17182	9460%	598685	225614	165%
		α baixo e β alto	5732126	53695	10862%	3675124	229446	1504%
	Longo	α e β altos	5771549	52499	11003%	3521034	229632	1435%
		α e β baixos	2672065	45243	5862%	1199566	227384	428%
		α alto e β baixo	2655177	44314	5945%	1176041	227137	418%
	Longo	α baixo e β alto	26872135	518956	5094%	12120538	254984	4664%
		α e β altos	27474098	522884	5173%	12606735	257574	4797%
	Curto	α e β baixos	1297545	15376	9289%	537109	225391	138%
		α alto e β baixo	2547749	15359	19440%	521045	225404	131%
		α baixo e β alto	4189854	21736	20558%	2123575	226281	841%
	Longo	α e β altos	5980109	21923	28358%	2155191	226555	853%
Variado	Longo	α e β baixos	2657163	21407	12994%	1106525	226789	387%
		α alto e β baixo	3161390	21171	15792%	1089796	226730	382%
		α baixo e β alto	26444347	124807	21497%	11201403	248954	4399%
	Curto	α e β altos	26792470	123502	21645%	11913785	247237	4733%
		α e β baixos	1242119	15678	8715%	556651	225431	147%
		α alto e β baixo	2095611	15842	13836%	558477	225468	148%
	Longo	α baixo e β alto	4992180	30787	16868%	2872851	228099	1161%
		α e β altos	5727603	27943	21531%	2817209	227418	1139%
	Curto	α e β baixos	2543903	25875	10218%	1119558	226860	394%
		α alto e β baixo	3107388	27735	11326%	1162956	227177	412%
		α baixo e β alto	27740461	234427	11757%	12022669	254650	4623%
	Longo	α e β altos	27740025	244684	11310%	12184890	255015	4684%

Tabela 5-12 – Média dos resultados dos experimentos computacionais com CPLEX e métodos implementados para instâncias com 100 itens e itens “grandes” (classe 9)

Prazo	Setup	Custos	CPLEX			Métodos propostos		
			UB	LB	gap	UB	LB	gap
Curto	Curto	α e β baixos	1299708	77704	2689%	1074727	685378	57%
		α alto e β baixo	1568589	75578	3294%	1066567	685366	56%
		α baixo e β alto	5574683	110110	6065%	4933390	691319	614%
	Longo	α e β altos	5878631	111828	6287%	5100949	690900	638%
		α e β baixos	2725143	96408	3563%	2225822	687645	225%
		α alto e β baixo	2663560	96119	3558%	2146662	687828	213%
	Longo	α baixo e β alto	27651833	544097	5020%	23197095	734887	3049%
		α e β altos	29061874	565690	5076%	23882865	731496	3173%
	Curto	α e β baixos	1299495	80173	2864%	972076	685059	42%
		α alto e β baixo	2555271	76032	4736%	982742	685052	44%
		α baixo e β alto	4284307	75998	9149%	3590059	685995	425%
	Longo	α e β altos	5907871	82429	13255%	3478857	685811	409%
Variado	Longo	α e β baixos	2539575	72650	6015%	2082383	686806	203%
		α alto e β baixo	3866022	78587	7399%	2068147	687017	201%
		α baixo e β alto	26182792	182675	15615%	21793311	718857	2941%
	Curto	α e β altos	27505090	205593	14723%	21406194	720840	2877%
		α e β baixos	1292521	81081	2749%	1018988	685134	49%
		α alto e β baixo	2431368	69423	7840%	1027286	685124	51%
	Longo	α baixo e β alto	5024972	81460	9131%	4368559	687495	536%
		α e β altos	5634233	89065	9551%	4126604	687362	502%
		α e β baixos	2548254	77320	5230%	2077344	686977	203%
	Curto	α alto e β baixo	3286864	79461	7192%	2041111	687481	198%
		α baixo e β alto	27773047	262744	10735%	23314888	729204	3099%
		α e β altos	30507506	303534	10235%	24644957	733387	3286%

6. DISCUSSÃO

O objetivo deste capítulo é desenvolver de maneira estruturada as principais contribuições que a metodologia aplicada neste trabalho fazem ao problema proposto. Para isso, a discussão foi separada em três núcleos distintos:

- Um primeiro ponto é a formulação do problema e a dificuldade do combinatório gerado, analisando seu reflexo na qualidade dos *gaps* gerados, tanto pelo CPLEX quanto pelos métodos propostos;
- Posteriormente, será conduzida uma análise da comparação dos resultados obtidos nestes dois caminhos, detalhando a superioridade entre eles de acordo com os diversos fatores elencados no procedimento experimental;

6.1. Formulação e dificuldade do problema

A formulação matemática apresentada no capítulo 4 retrata o problema de programação de operações de corte em ambiente de máquina única, visando a minimização de atrasos e adiantamentos ponderados, conforme o objetivo do trabalho de explorar uma junção entre problemas de *scheduling* e corte e empacotamento.

Dentre os pontos a destacar para a formulação está seu formato em que a entidade *bin* não é considerada, de maneira que o número de variáveis binárias do problema conjunto não supera o número de variáveis binárias dos modelos somados. Entretanto, como são decisões de naturezas diferentes, não foi possível obter uma redução deste número por alguma forma de sobreposição de funções.

Da mesma maneira, o número de restrições também não apresentou sinergia, porém não houve aumento do número de restrições. A restrição de precedência dos *bins* no modelo inicial foi suprimida, porém o *scheduling* dos itens precisa do dobro de restrições do modelo original. Importante notar que para este diagnóstico não foram computadas as restrições do conjunto (3.23) do modelo, já que esta igualdade só existe para uma melhor apresentação e compreensão do modelo, podendo a variável t_i ser, naturalmente, substituída pelo lado direito da igualdade nas restrições (3.21) e (3.22).

É intuitivo que o modelo resultante seja NP-difícil, uma vez que parte de dois outros com a mesma característica. De fato, os resultados dos experimentos computacionais mostraram que, mesmo para as menores instâncias (20 itens), não é possível obter a solução ótima. Mais do que isto, os *gaps* obtidos, tanto com o *software* de mercado (CPLEX), quanto com os métodos propostos no capítulo 5 foram extremamente elevados. A tabela 6.1 mostra estes valores, em média, discriminados mediante os fatores de geração de instâncias no plano experimental.

Tabela 6-1 – *Gaps* obtidos nos diferentes cenários experimentais

Critério	Classificação	<i>Gaps</i>	
		CPLEX	Métodos propostos
Classe	7	6053,9%	942,6%
	8	6665,8%	1124,0%
	9	2901,5%	569,9%
	10	6711,9%	1149,9%
Porte	20	186,5%	227,1%
	60	4331,5%	976,5%
	100	12231,7%	1636,2%
Custos	α e β baixos	4047,3%	187,3%
	α alto e β baixo	5493,4%	185,8%
	α baixo e β alto	6068,6%	1689,5%
	α e β altos	6723,8%	1724,0%
Setup	Curto	6111,5%	402,4%
	Longo	5055,0%	1490,8%
Prazo	Curto	3444,0%	1030,2%
	Longo	7591,8%	877,5%
	Variado	5714,0%	932,2%
Média		5583,3%	946,6%

Os *gaps* progressivamente maiores à medida que o porte das instâncias aumenta é algo bastante razoável e esperado. Cabe destaque aqui, entretanto, a diferença significativamente menor¹ entre os *gaps* obtidos na classe 9 em relação aos demais. Trata-se da classe com itens grandes (maiores que a metade do tamanho do *bin* em ambas dimensões). Fica claro que,

¹ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.1 e E.2 para o CPLEX e Tabelas E.3 e E.4 para os métodos propostos

neste caso, o problema passa a ser de *scheduling* simplesmente, uma vez que cada item tem que estar isolado em uma única placa.

As demais diferenças marcantes entre *gaps* parecem ser mais peculiaridades dos métodos propostos nas condições de custo de atraso baixo e *setup* curto, do que propriamente propriedades das instâncias, uma vez que os resultados do CPLEX não são fielmente consistentes com qualquer inferência a este respeito. O custo de atraso abaixo ainda apresenta uma ligeira diferença em ambas formas de resolução, o que induz à reflexão de uma possibilidade de melhores limitantes inferiores (já que penalidades de atraso não apresentam limitações, enquanto por adiantamento são truncadas no instante zero).

Retomando os números gerais, reforça-se que são *gaps* bastante dilatados. Enquanto o CPLEX encontrou soluções em média distantes do *lower bound* em quase 56 vezes, os métodos propostos ainda resultaram em uma distância média de quase 10 vezes esse valor. Embora seja uma melhoria expressiva, destaca-se mais, aqui, um indicativo do quanto o problema é de difícil resolução.

6.2. Comparação entre os métodos propostos e o *software* de mercado (CPLEX)

A título de compilação e, conseqüentemente, uma melhor apreciação dos resultados obtidos e mostrados no capítulo 5, a Tabela 6.2 sumariza a comparação percentual entre o método proposto e os resultados obtidos no *software* de mercado, conforme a equação (5.2).

Os resultados da comparação em cada fator de análise proposto no plano experimental são vistos separadamente, de forma a apreciar a robustez da superioridade do método.

Reforça-se aqui que existe superioridade do método proposto quando a comparação percentual é negativa no caso do *upper bound* (método propôs um limitante superior mais baixo); e positiva no caso do *lower bound* (método propôs um limitante inferior mais baixo). Seguindo o mesmo raciocínio, há superioridade do método na comparação percentual do *gap* quando esta é negativa.

Tabela 6-2 – Comparação percentual entre os métodos propostos no capítulo 4 e os resultados obtidos no CPLEX

Critério	Classificação	UB %	LB %	gap %
Classe	7	-19,7%	396,9%	-30,1%
	8	-8,8%	480,5%	-18,4%
	9	-0,5%	409,7%	-26,6%
	10	-22,3%	273,4%	-2,9%
Porte	20	47,6%	81,4%	96,4%
	60	-38,4%	343,9%	-72,7%
	100	-47,7%	745,0%	-82,2%
Custos	α e β baixos	-13,8%	597,1%	-75,1%
	α alto e β baixo	-19,8%	606,2%	-76,8%
	α baixo e β alto	-7,5%	182,4%	39,2%
	α e β altos	-10,1%	174,7%	34,6%
Setup	Curto	-15,3%	567,4%	-68,0%
	Longo	-10,3%	212,8%	29,0%
Prazo	Curto	-10,0%	283,6%	3,8%
	Longo	-16,1%	476,7%	-39,1%
	Variado	-12,3%	410,0%	-23,3%
Média		-12,8%	390,1%	-19,5%

A diferença observada entre os valores de limitantes superiores com o CPLEX e a heurística construtiva proposta, que levam à seguinte comparação percentual é estatisticamente significativa². Testados de maneira análoga, os valores de limitantes inferiores também apresentaram diferença estatisticamente relevante³.

De maneira análoga, foi computada a porcentagem das instâncias em que os métodos propostos foram superiores ao CPLEX. Aqui não se trata de mensurar o quanto superior o método foi, mas, em quantos casos isto aconteceu.

Uma observação resumida das tabelas, na média final das 1.152 instâncias mostra que a combinação dos métodos propostos com 60 segundos de tempo computacional apresenta um resultado final superior ao CPLEX, mesmo com 600 segundos rodados por instância. O *gap* deste trabalho é 19,5% inferior ao obtido pelo CPLEX, sendo inferior em 83,2% das

² Teste t de Student assumindo variâncias desiguais e amostras não pareadas com valor-p inferior a 0,0001 (ou seja, $p < 0,05$). Detalhes do teste no apêndice E, Tabela E.5.

³ Teste t de Student assumindo variâncias desiguais e amostras não pareadas com valor-p inferior a 0,0001 (ou seja, $p < 0,05$). Detalhes do teste no apêndice E, Tabela E.6.

instâncias. Fica claro também que grande parte da superioridade das heurísticas propostas advém da melhor qualidade do limitante inferior, que é melhorado em 81,7% dos casos, sendo em média 390,1% maior do que aquele obtido pelo CPLEX. O resultado para o *upper bound* já é mais discreto, sendo superior em pouco mais que a metade das instâncias (63,0%) apenas e tendo um resultado de diminuição de 12,8%.

Tabela 6-3 – Porcentagem de instâncias em que os métodos propostos foram superiores ao CPLEX

Critério	Classificação	UB	LB	GAP
Classe	7	66,7%	82,3%	86,1%
	8	66,7%	83,0%	83,7%
	9	52,1%	84,0%	79,9%
	10	66,7%	77,4%	83,0%
Porte	20	0,3%	69,0%	58,9%
	60	88,8%	84,9%	94,0%
	100	100,0%	91,1%	96,6%
Custos	α e β baixos	62,5%	98,3%	96,2%
	α alto e β baixo	61,5%	97,9%	95,1%
	α baixo e β alto	64,2%	65,3%	70,5%
	α e β altos	63,9%	65,3%	70,8%
Setup	Curto	61,5%	96,9%	91,8%
	Longo	64,6%	66,5%	74,5%
Prazo	Curto	63,8%	75,0%	73,4%
	Longo	62,2%	88,8%	88,5%
	Variado	63,0%	81,3%	87,5%
Média		63,0%	81,7%	83,2%

Entretanto, a análise dos diferentes fatores de análise criados na geração das instâncias leva a conclusões mais profundas da comparação entre os métodos.

6.2.1. Classes (tamanho dos itens)

Conforme analisado previamente (ver apêndice A, Tabelas A.1, A.2, A.3 e A.4), existe diferença estatística significativa entre os resultados obtidos pelos métodos propostos e pelo CPLEX entre classes. Entretanto, vale destacar aqui que existe ainda uma diferença também estatisticamente significativa na comparação percentual entre os métodos, tanto no caso dos *upper bounds*⁴, quanto no caso dos *lower bounds*⁵.

Fica pronunciada a diferença existente do desempenho do limitante superior na classe 9 (itens com dimensões maiores do que a metade do *bin* em ambos os sentidos) e do limitante inferior na classe 10 (itens com dimensões menores do que a metade do *bin* em ambos os sentidos).

A primeira diferença pode ser resultado de uma simplificação do problema, que praticamente reduz o combinatório a um problema de programação de itens (*scheduling*) de tal forma que a heurística desenhada para a junção dos problemas aproveita-se menos de suas características físicas particulares. Reforça esta hipótese o fato de que os resultados do limitante superior na classe 9 são consistentemente inferiores de acordo com o número de itens, como pode ser visto na Tabela 6.4.

Tabela 6-4 – Comparação percentual dos limitantes superiores quebrados por classe e porte, com destaque para a classe 9 (“itens grandes”)

		Porte			
		20	60	100	Média
Classe	7	47,4%	-50,6%	-55,8%	-19,7%
	8	63,1%	-41,9%	-47,6%	-8,8%
	9	25,3%	-1,9%	-24,9%	-0,5%
	10	54,7%	-59,1%	-62,4%	-22,3%
	Média	47,6%	-38,4%	-47,7%	-12,8%

Quanto à diferença de desempenho do *lower bound* na classe 10 (“itens pequenos”), existe uma provável influência do *lower bound* do problema de *bin packing* associado, que é colocado no início do algoritmo. Com isto, em instâncias em que se consegue melhorar a

⁴ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.7 e E.8.

⁵ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.9 e E.10.

estimativa mínima de *bins* necessários, o *lower bound* para o problema de programação de itens empacotados torna-se melhor ajustado.

6.2.2. Porte da instância (número de itens)

Analisando a comparação percentual entre os métodos propostos e o CPLEX agrupando as instâncias por seu porte, ou seja, o número de itens, também verifica-se que existe diferença estatisticamente significativa, tanto no caso dos *upper bounds*⁶, quanto no caso dos *lower bounds*⁷.

Os resultados deixam bastante claro que o aproveitamento das características físicas do problema aumenta quanto maior a sua complexidade em termos computacionais. Isto é particularmente verdadeiro no caso do limitante superior, em que raros foram os casos de vitória da heurística construtiva (0,3% das instâncias) em relação ao CPLEX para 20 itens; ao passo que, para 100 itens houve melhorias em 100% das instâncias, sendo em média 47,7% melhorados.

Mais do que a conclusão geral pela média, observa-se que em todos os cenários propostos essa afirmação é consistente. O Gráfico 6.1 ilustra que existe uma evolução constante do limitante superior conforme o número de itens aumenta, havendo uma grande melhoria de 20 para 60 itens, sendo menos acelerada de 60 para 100 (exceto para a classe 9 de tamanho de itens, que já foi explorada anteriormente e tem evolução quase linear).

⁶ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.11 e E.12.

⁷ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.13 e E.14.

Gráfico 6-1 – Comparação entre *upper bounds* nos diferentes cenários experimentais, destacando o número de itens

A mesma melhoria progressiva acontece no *lower bound*, o que pode ser observado no Gráfico 6.2. Em geral, o limitante inferior foi melhorado 390,1%, porém segue a disparidade com relação ao número de itens, sendo apenas 81,4% para instâncias com 20 itens, 343,9% para instâncias com 60 itens e 745% para instâncias com 100 itens, em média. Aqui existe uma evolução diferente do *upper bound* na medida em que as melhorias vão sendo progressivamente maiores. Isto ocorre, provavelmente, porque com o aumento da quantidade de itens, o CPLEX precisa de mais tempo para encontrar sequer uma solução inicial, sendo muito custoso avançar no combinatório de forma a melhorar os limitantes inferiores.

Gráfico 6-2 – Comparação entre *lower bounds* nos diferentes cenários experimentais, destacando o número de itens

6.2.3. Custos relativos de adiantamento, atraso e matéria-prima

Quando analisadas sob o corte da relevância dos custos de adiantamento, atraso e matéria-prima, novamente identifica-se que existe diferença estatística significativa entre a comparação percentual entre os métodos propostos e o CPLEX. Isto, como nos outros casos, foi verdadeiro tanto para os *upper bounds*⁸, quanto para os *lower bounds*⁹.

⁸ ANOVA de fator único, com valor-p de 0,01578 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.15 e E.16.

⁹ ANOVA de fator único, com valor-p inferior a 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.17 e E.18.

Fica bastante notório que os altos custos de atraso (β_i) levam a piores resultados relativos dos métodos propostos, tanto quando observamos a heurística construtiva, quanto a heurística de geração de *lower bound*.

A Tabela 6.5 evidencia esta comparação para os limitantes superiores. Fica claro que a heurística se sobressai com custos de atraso mais baixos, tanto em média quanto em face a ambos os cenários de custos de adiantamento. A provável origem deste ponto negativo está no fato de que não existe um tratamento mais cuidadoso em caso de sobreposição temporal de lotes programados pela heurística, o que pode ser um ponto de melhoria. Quando a data ideal encontrada do *bin* leva a uma sobreposição temporal, a heurística aloca o lote logo na primeira data disponível, ou seja, realoca-o sempre à frente, levando a mais atrasos.

Tabela 6-5 – Comparação percentual dos *upper bounds* destacando os custos unitários de atraso e adiantamento

		Custo de adiantamento (α)		
		Alto	Baixo	Média
Custo de atraso (β)	Alto	-10,1%	-7,5%	-8,8%
	Baixo	-19,8%	-13,8%	-16,8%
	Média	-14,9%	-10,7%	-12,8%

Os custos de adiantamento inferiores também levam a melhores resultados da heurística construtiva consistentemente, como pode ser visto também na tabela 6.5. Entretanto, os ganhos aqui são proporcionalmente menores. A explicação para tal parece residir no mesmo ponto, sendo limitada pelo fato de que os custos com adiantamento são naturalmente truncados pelo instante zero, independente do número de itens e distribuição de prazos.

No caso dos *lower bounds*, que estão detalhados segundo custos unitários de atraso e adiantamento na Tabela 6.6, o comportamento é semelhante, porém de maneira ainda mais acentuada para os custos de atraso. Verifica-se que a superioridade do método criado é 3 vezes maior percentualmente quando os custos de atraso são mais baixos. A hipótese de origem desta diferença é a mesma da anterior: o método trata sobreposições temporais com uma programação mais tardia do lote em relação à sua data ideal.

Tabela 6-6 – Comparação percentual dos *lower bounds* destacando os custos unitários de atraso e adiantamento

		Custo de adiantamento (α)		
		Alto	Baixo	Média
Custo de atraso (β)	Alto	174,7%	182,4%	178,6%
	Baixo	606,2%	597,1%	601,6%
	Média	390,4%	389,8%	390,1%

Cabe destacar aqui que, no caso dos custos de adiantamento, não existe diferença significativa entre os casos em que o custo de adiantamento é alto ou baixo¹⁰.

6.2.4. Setup

A comparação percentual entre os métodos propostos e o CPLEX agrupando as instâncias pela extensão do *setup* apresentou diferença estatística significativa apenas no caso dos *lower bounds*¹¹, enquanto que para os *upper bounds*¹² esta diferença não foi verificada.

A conclusão de inexistência de diferença significativa na comparação de *upper bounds* entre *setups* longos e curtos pode ser ilustrada através do Gráfico 6.3, no qual é possível verificar a proximidade de valores nos cenários deste critério.

O Gráfico 6.4, por sua vez, mostra que – apesar de haver uma diferença que se faz notória entre instâncias com *setups* curtos e longos, o formato das comparações percentuais em relação aos demais fatores tem o mesmo desenho para ambos. Isto significa que trata-se em um fator pouco relevante na comparação, cuja variação encontrada mais se relaciona a outros critérios do que a ele próprio.

¹⁰ Teste t de Student assumindo variâncias desiguais e amostras não pareadas com valor-p de 0,9842 (ou seja, $p > 0,05$). Detalhes do teste no apêndice E, Tabela E.19.

¹¹ Teste t de Student assumindo variâncias desiguais e amostras não pareadas com valor-p de 0,0860 (ou seja, $p > 0,05$). Detalhes do teste no apêndice E, Tabela E.20.

¹² Teste t de Student assumindo variâncias desiguais e amostras não pareadas com valor-p $< 0,0001$ (ou seja, $p < 0,05$). Detalhes do teste no apêndice E, Tabela E.21.

Gráfico 6-3 – Comparação percentual do *upper bound* entre instâncias com *setups* curtos e longos sob a ótica de outros fatores experimentais

Gráfico 6-4 – Comparação percentual do *lower bound* entre instâncias com *setups* curtos e longos sob a ótica de outros fatores experimentais

Ainda observando este critério, podemos concluir que em instâncias de *setup* curto, a geração de *lower bounds* favorece o método proposto. Mais uma vez, a provável causa deve ser o tratamento da sobreposição temporal, haja vista o melhor desempenho com prazos longos e variados em face ao pior desempenho com prazos curtos.

6.2.5. Datas de entrega

Finalmente, com relação às datas de entrega, foram geradas simulando prazos de execução curtos, longo e variados. A comparação percentual entre os métodos propostos e o CPLEX, agrupando as instâncias por este critério, apresentou diferença estatística significativa apenas no caso dos *lower bounds*¹³, enquanto que para os *upper bounds*¹⁴ esta diferença não foi verificada novamente.

Mais uma vez, analisando os resultados do limitante superior em face à outros fatores, não existe nenhuma interação aparente do desempenho de um certo cenário. Assim, a constatação do equilíbrio por comparação estatística das médias é consistente em diferentes cenários de outros fatores, conforme ilustra o Gráfico 6.5.

Gráfico 6-5 – Comparação percentual do *upper bound* entre instâncias com datas de entrega próximas, distantes ou variadas sob a ótica de outros fatores experimentais

¹³ ANOVA de fator único, com valor-p de 0,2134 (ou seja, maior do que 0,05). Detalhes do teste no apêndice E, Tabelas E.22 e E.23.

¹⁴ ANOVA de fator único, com valor-p < 0,0001 (ou seja, menor do que 0,05). Detalhes do teste no apêndice E, Tabelas E.24 e E.25.

Já no *lower bound*, a diferença de superioridade do método foi encontrada, sendo mais pronunciada para datas de entrega mais longas, seguida de variadas e, por fim, as curtas. Novamente, não há indício da interação deste fator com outros, haja vista que o formato da curva de aumento ou diminuição de desempenho comparativo é igual em todos os outros cenários, como pode ser observado no Gráfico 6.6.

Gráfico 6-6 – Comparação percentual do *lower bound* entre instâncias com datas de entrega próximas, distantes ou variadas sob a ótica de outros fatores experimentais

7. CONCLUSÕES E EXTENSÕES

O presente trabalho tem como contribuição inicial o aprofundamento da junção entre problemas de corte e empacotamento e programação da produção. Embora o tema já tenha sido abordado na literatura de maneira indireta, trata-se de uma abordagem inovadora na medida em que modela as restrições inerentes a ambos problemas e considera a minimização de custos globais.

No que tange ao objetivo de modelar o problema, foi proposto no capítulo 3 um modelo de programação linear inteira mista que utiliza-se de ideias advindas dos dois campos da literatura para um modelo conjunto. Ainda que não gere sinergias entre os dois modelos, diminuindo o número de variáveis binárias ou restrições, também não necessita de um aumento da soma destes para unir ambos.

Como o modelo proposto é NP-hard, existe uma grande dificuldade de resolução para portes próximos de problemas reais. Isso fica reforçado no capítulo 6, onde são mostrados os grandes *gaps* encontrados para o problema, mesmo com 10 minutos de tempo para o CPLEX.

Desta forma, fez-se necessário um avanço do trabalho no campo da proposição de métodos de solução para o problema, bem como a exploração das condições geradoras de maiores dificuldades de resolução.

Em relação a este segundo objetivo de propor uma melhor resolução para o problema, o trabalho dividiu-se em dois esforços:

- Uma heurística construtiva para encontrar uma solução melhorada em tempo computacional razoável;
- Um método de geração de limitante inferior, de forma a poder avaliar melhor esta heurística construtiva, haja vistos os enormes *gaps* que o *software* de mercado encontra.

O desenvolvimento de ambos métodos foi apoiado sobre características físicas do problema estudado. Destaca-se neste trabalho a proposição de 2 propriedades específicas deste problema que contribui para a melhoria da qualidade da geração de soluções e limitantes. Ambas propriedades dizem respeito ao posicionamento temporal de um conjunto de itens previamente designados a um lote. A primeira propriedade prova que este instante coincide com uma das datas de entrega dos itens que formam o lote, desde que não haja nenhum

restrição temporal que impeça esta programação (como, por exemplo, a necessidade de iniciar o lote antes do instante zero para tal; ou ainda de sobrepor-se ao tempo de processamento de outro lote). A segunda propriedade mostra que, dado um conjunto de itens que forma um lote, a função penalidade por desvio de datas é uma linear por partes que possui concavidade sempre positiva, ou seja, apresenta um mínimo global, antes do qual esta função é sempre decrescente em relação ao tempo; e a partir do qual a função penalidade é crescente em relação ao tempo.

Com estas propriedades, os métodos propostos se reduzem a um problema de combinação de itens em lotes, o que não faz dele um problema simples, mas diminui sensivelmente seu combinatório. Entretanto, vale ressaltar que esta redução, mesmo levando em conta estas propriedades, é uma aproximação, dado que, mesmo tendo os instantes ideais de programação de um conjunto de itens, não há garantia de não-sobreposição em relação a outros lotes.

Ainda que esta aproximação tenha sido feita, os resultados dos métodos proposto foram bastante interessantes. Se por um lado continuaram resultando em *gaps* muito grandes, insatisfatórios para serem consideradas boas soluções, por outro propiciaram uma redução importante do que o CPLEX pode proporcionar. Sobretudo no que diz respeito aos limitantes inferiores, o método heurístico desenvolvido aumenta-o em cerca de 4 vezes em média, sendo até 53 vezes em casos extremos de certas instâncias.

A extensa massa de instâncias geradas teve como objetivo explorar características técnicas da realidade que podem influenciar na dificuldade de resolução do problema, assim como mapear as forças e fraquezas de um determinado método de resolução. Os fatores experimentais considerados foram: tamanho médio dos itens (4 níveis), porte da instância em número de itens (3 níveis), datas de entrega (3 níveis), tempo de *setup* (2 níveis) e custo unitário de atraso e adiantamento em relação à matéria-prima (4 níveis). Para cada combinação de fatores, foram geradas 4 repetições, de forma que a massa total de testes foi de 1.152 experimentos.

Em média, a heurística construtiva proposta é 12,8% superior à solução gerada pelo CPLEX, sendo superior em 63% dos 1.152 casos testados. Já a heurística de geração de *lower bound* melhora em média 390% o resultado do CPLEX, sendo superior em quase 82% das instâncias testadas. Vale ressaltar que enquanto o CPLEX teve 600 segundos de tempo de

processamento como limite, o método proposto foi submetido apenas a 60 segundos de tempo computacional.

Em relação aos resultados nos diferentes cenários, várias vezes ficou claro que a falta de tratamento da sobreposição temporal dos lotes foi um ponto fraco dos métodos propostos. Reforçam esta conclusão os resultados piores para prazos curtos, *setups* longos e altos custos unitários de atraso. Outro ponto reforçado é o aumento da diferença de *performance* em relação ao tamanho da instância. Isto mostra que sua lógica geral é forte, porém carece de mecanismos de pós-otimização que dêem um ajuste fino à solução encontrada. Fica claro que para instâncias de 20 itens o resultado da heurística construtiva é bastante ruim, sendo o principal detrator de um bom desempenho médio.

O trabalho abre campo para uma infinidade de melhorias em relação à métodos heurísticos. Não se pode perder de vista que, ainda que os métodos propostos tenham melhorado o enorme *gap* médio de 5.583% gerado pelo CPLEX, ainda há espaço para muitas melhorias, visto que só se chegou a um *gap* médio de 947%, que ainda é bastante insatisfatório e tem espaço para largos estreitamentos.

Em relação ao próprio método criado, dois pontos principais são oportunidades de melhorias para o seu desempenho: um tratamento melhor para o caso de sobreposição temporal do tempo de execução dos lotes, quando as datas encontradas levarem a esta condição; e passos de pós-otimização na heurística construtiva para melhorar o ajuste fino da solução encontrada, haja visto o pior desempenho em instâncias com menos itens.

REFERÊNCIAS¹⁵

- ALLEN, S. D.; BURKE, E. K.; KENDALL, G. A hybrid placement strategy for the three-dimensional strip packing problem. **European Journal of Operational Research**, v. 209, p. 219-227, 2011.
- ALVES, C.; CARVALHO, J. M. V. New integer programming formulations and an exact algorithm for the ordered cutting stock problem. **Journal of the Operational Research Society**, v.59, p. 1520-1531, 2008.
- ARENALES, M.; MORABITO, R. An AND/OR-graph approach to the solution of two-dimensional non-guillotine cutting. **European Journal of Operational Research**, v. 87, p. 599-617, 1995.
- AZIZOGLU, M.; KNODAKCI, S.; KIRCA, O. Bicriteria scheduling problem involving total tardiness and total earliness penalties. **International Journal of Production Economics**, v. 23, p. 17-24, 1991.
- BEASLEY, J. E. Algorithms for unconstrained two-dimensional guillotine cutting. **Journal of the Operational Research Society**, v. 36, p. 297-306, 1985.
- BICHOFF, E. E.; WÄSCHER, G. Cutting and Packing. Editorial. **European Journal of Operational Research**, v. 84, p. 503-505, 1995.
- BISKUP, D.; FELDMANN, M. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. **Computers & Operations Research**, v. 28, p. 787-801, 2001.
- BORTFELDT, A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. **European Journal of Operational Research**, v. 172, p. 814-837, 2006.
- BOSCHETTI, M. A.; MINGOZZI, A. The two-dimensional Finite Bin Packing Problem. Part I: New lower bounds for the oriented case. **4OR**, v.1, p. 1-16, 2002.
- BOSCHETTI, M. A.; MINGOZZI, A.; HADJICONSTANTINO, E. New upper bounds for the two-dimensional orthogonal non-guillotine cutting stock problem. **IMA Journal of Management Mathematics**, v.13, p. 95-119, 2002.

¹⁵ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

- CARLIER, J.; CLAUTIAUX, F.; MOUKRIM, A. New reduction procedures and lower bounds for the two-dimensional bin packing problem with fixed orientation. **Computers & Operations Research**, v. 34, p. 2233-2250, 2007.
- CHEN, C. S.; LEE, S. M.; SHEN, Q. S. An analytical model for the container loading problem. **European Journal of Operational Research**, v. 80, p. 68-76, 1995.
- CHRYSSOLOURIS, G. PAPAKOSTAS, N., MOURTZIS, D. A decision-making approach for nesting scheduling: A textile case. **International Journal of Production Research**, v. 38: 17, pp.4555 — 4564, 2000.
- CLARKE, G.; WRIGHT, J. W. Scheduling of vehicles from central depot to number of delivery points. **Operations Research**, v. 12, p. 568-581, 1964.
- CLAUTIAUX, F.; CARLIER, J.; MOUKRIM, A. A new exact method for the two-dimensional orthogonal packing problem. **European Journal of Operational Research**, v. 183, p. 1196-1211, 2007.
- CONWAY, R. W. Priority Dispatching and Job Lateness in a Job Shop. **Journal of Industrial Engineering**, v. 16, p. 228-237, 1965.
- CONWAY, R. W., MAXWELL, W. L., MILLER, L. W. **Theory of Scheduling**. Reading: Addison-Wesley, 1967.
- CRAIGHEAD, C. W.; LAFORGE, R. L. Taxonomy of information technology adoption patterns in manufacturing firms. **International Journal of Production Research**, v. 41, p. 2431-2449, 2003.
- DOLATABADI, M.; LODI, A.; MONACI, M. Exact algorithms for the two-dimensional guillotine knapsack. **Computers & Operations Research**, v. 39, p. 48-53, 2012.
- DOWNSLAND, K. A.; DOWNSLAND, W. B. Packing Problems. **European Journal of Operational Research**, v. 56, p.2-14, 1992.
- DOWNSLAND, K. A.; DOWNSLAND, W. B. Solution approaches to irregular nesting problems. **European Journal of Operational Research**, v. 84, p.506-521, 1995.
- DYCKHOFF, H.; WÄSCHER, G. Cutting and packing. **European Journal of Operational Research**, v. 44, p. 145–159, 1990.
- DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, v. 44, p. 145–159, 1990.

- FEKETE, S. P.; SCHEPERS, J. New Classes of Lower Bounds for Bin Packing Problems. **Integer Programming and Combinatorial Optimization (IPCO 98)**, v. 1412, p. 257-270, 1998.
- FEKETE, S. P.; SCHEPERS, J. On More-Dimensional Packing I: Modeling. **ZPR Technical Report**, v. 97-288, 2000.
- FEKETE, S. P.; SCHEPERS, J. On More-Dimensional Packing II: Bounds. **ZPR Technical Report**, v. 97-289, 2000.
- FEKETE, S. P.; SCHEPERS, J. On More-Dimensional Packing III: Exact Algorithms. **ZPR Technical Report**, v. 97-290, 2000.
- FEKETE, S. P.; SCHEPERS, J.; VAN DER VEEN, J. C. An Exact Algorithm for Higher-Dimensional Orthogonal Packing. **Operations Research**, v. 55, n. 3, p. 569-587, 2007.
- FINK, A.; VOß, S. Applications of modern heuristic search methods to pattern sequencing problems. **Computers & Operations Research**, v.26, p. 17-34, 1999.
- FOERSTER, H.; WÄSCHER, G. Simulated annealing for order spread minimization in sequencing cutting patterns. **European Journal of Operational Research**, v. 110, p. 272-281, 1998.
- GILMORE, P. C.; GOMORY, R. E. Multistage cutting stock problems of 2 and more dimensions. **Operations Research**, v. 13, p. 94-120, 1965.
- GOLDEN, B. L. Approaches to the Cutting-Stock Problem. **AIIE Transactions**, v. 8, p. 256-274, 1976.
- GRAMANI, M. C. N., (2001) “Otimização do processo de cortagem acoplado ao planejamento da produção”. Tese de Doutorado, FEEC – UNICAMP.
- GUPTA, J. N. D.; STAFFORD, E. F. Flowshop scheduling research after five decades. **European Journal of Operational Research**, v. 169, p. 699-711, 2006.
- HADJICONSTANTINOU, E.; CHRISTOFIDES, N. An exact algorithm for general, orthogonal, two-dimensional knapsack problems. **European Journal of Operational Research**, v. 83, p. 39-56, 1995.
- HERRMANN, J. W. **Handbook of production scheduling**. NewYork: Springer, 2006. 318 p.

HIFI, M., KACEM, I., NÈGRE, S., WU, L. A Linear Programming Approach for the Three-Dimensional Bin-Packing Problem. **Electronic Notes in Discrete Mathematics**, v. 36, pp.993-1000, 2010.

KANTOROVICH, L. V. Mathematical methods of organizing and planning production. **Management Science**, v. 6, p. 366-422, 1960.

KELLEY, J., WALKER, M. Critical-Path Planning and Scheduling. Proceedings of the Eastern Joint Computer Conference, 1959.

KIM, Y. D.; YANO, C. A. Minimizing mean tardiness and earliness in single-machine scheduling problems with unequal due dates. **Naval Research Logistics**, v. 41, p. 913-933, 1994.

LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G. Recent Developments in Deterministic Sequencing and Scheduling: A Survey in Deterministic and Stochastic Scheduling, Dempster, Lentra and Rinnooy Kan (eds.), pp. 35-74, Reidel, Dordrecht, 1982.

LEE, C. Y.; CHOI, J. Y. A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights. **Computers & Operations Research**, v. 22, n. 8, p. 857-869, 1995.

LEE, C. Y. Machine Scheduling with Availability Constraints. Chapter 22 in Handbook of Scheduling, J. Y. T. Leung (ed.), Chapman and Hall/CRC, Boca Raton, Florida, 2004.

LEMOS, F. K., SANTORO, M. C. Sequenciamento de ordens com operações em lotes bidimensionais não-guilhotinados considerado atendimento de datas e ocupação da matéria-prima. **Anais do XLII Simpósio Brasileiro de Pesquisa Operacional**, 2010.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, VLSI design, and flexible machines. **Computers & Operations Research**, v. 29, p. 1759-1772, 2002.

LIU, Y.; CHU, C.; WANG, K. A new heuristic algorithm for a class of two-dimensional bin-packing problems. **International Journal of Advanced Technology**, v. 57, p. 1235-1244, 2011.

LODI, A.; MARTELLO, S.; VIGO, D. Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. **INFORMS Journal on Computing**, v. 11, n. 4, p. 345-357, 1999.

- LODI, A.; MARTELLO, S.; VIGO, D. Recent advances on two-dimensional bin packing problems. **Discrete Applied Mathematics**, v. 123, p. 379-396, 2002.
- M'HALLAH, R. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. **Computers & Operations Research**, v. 29, p. 1759-1772, 2002.
- MARTELLO, S.; VIGO, D. Exact solution of the two-dimensional finite bin packing problem. **Management Science**, v. 44, p. 388-399, 1998.
- MIURA, M. **Modelagem heurística no problema de distribuição de cargas fracionadas de cimento**. 2008. 84 f. Dissertação (Mestrado em Engenharia) – Escola Politécnica, Universidade de São Paulo, São Paulo, 2008.
- MORABITO, R. Modelos de otimização para o problema de corte nas indústrias de papel e papelão e de móveis. **Gestão & Produção**, v. 1, n. 1, p. 59-76, 1994.
- MORABITO, R.; PUREZA, V. A heuristic approach based on dynamics programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. **Annals of the Operations Research**, v. 179, p. 297-315, 2010.
- MORABITO, R.; PUREZA, V. “*Modelagem e simulação*”, em: Metodologia de Pesquisa em Engenharia de Produção e Gestão de Operações, Paulo A. C. Miguel (ed), Editora Campus/Elsevier, Rio de Janeiro, 165-194, 2010.
- OLIVEIRA, J. F.; WÄSCHER, G. Cutting and Packing. **European Journal of Operational Research**, v. 183, p. 1106-1108, 2007.
- ORTMANN, F. G.; NTENE, N.; VAN VUUREN, J. H. New and improved level heuristics for the rectangular strip packing and variable-sized bin packing problems. **European Journal of Operational Research**, v. 203, p. 306-315, 2010.
- OVACIK, I. M.; UZSOY, R. **Decomposition methods for complex factory scheduling problems**. Norwell: Kluwer Academic Publishers, 1997. 232 p.
- PACHECO, R. F.; SANTORO, M. C. Proposta de Classificação Hierarquizada dos Modelos de Solução para o Problema de Job Shop Scheduling. **Gestão & Produção**, São Carlos, v. 6, n. 1, p. 1-15, 1999.
- PILEGGI, G. C. F.; MORABITO, R.; ARENALES, M. N. Heurísticas para os problemas de geração e sequenciamento de padrões de corte bidimensionais. **Pesquisa Operacional**, Rio de Janeiro, v. 27, p. 549-568, 2007.

- PINEDO, M. **Scheduling**: Theory, algorithms and systems. New Jersey: Prentice Hall, 1995. 378 p.
- PINEDO, M. **Scheduling**: Theory, algorithms and systems. New Jersey: Prentice Hall, 2008.
- POLDI, K. C. **O problema de corte de estoque multiperíodo**. 2007. 109 f. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2007.
- POLTRONIERI, S. C. **Otimização do processo de corte integrado à produção de bobinas – modelo e métodos de solução**. 2006. 118 f. Tese (Doutorado em Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2006.
- POTTS, C. N., KOVALYOV, M. Y. Scheduling with Batching: A Review. **European Journal of Operational Research**, Amsterdam, v. 120, pp. 228-249, 2000.
- RAGSDALE, C. T., ZOBEL, C. W. The ordered Cutting Stock Problem. **Decision Sciences**, v. 35:1, pp. 83-100, 2004.
- REINERTSEN, H., VOSSEN, T. W. M. The one-dimensional cutting stock problem with due dates. **European Journal of Operational Research**, v. 201, pp.701-711, 2010.
- SAWAYA, N. W., GROSSMANN, I. E., A cutting plane method for solving linear generalized disjunctive programming problems. **Computers and Chemical Engineering**, v. 29, 1891-1913, 2005.
- SCHALLER, J. A comparison of lower bounds for the single-machine early/tardy problem. **Computers & Operations Research**, v. 34, p. 2279-2292, 2007.
- SERAIRI, M.; HAOUARI, M. A Computational Study of Lower Bounds for the Two Dimensional Bin Packing Problem. **Electronic Notes in Discrete Mathematics**, v. 36, p. 891-897, 2010.
- SOURD, F. The continuous Assignment Problem and Its Application to Preemptive and Non-Preemptive Scheduling with Irregular Cost Functions. **INFORMS Journal on Computing**, v. 16, n. 2, p. 198-208, 2004.
- SOURD, F. Optimal timing of a sequence of tasks with general completion costs. **European Journal of Operational Research**, v. 165, p. 82-96, 2005.

- SOURD, F. New Exact Algorithms for One-Machine Earliness-Tardiness Scheduling. **INFORMS Journal on Computing**, v. 21, n. 1, p. 167-175, 2009.
- SOURD, F.; KEDAD-SIDHOUM, S. A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. **Journal of Scheduling**, v. 11, p. 49-58, 2007.
- TALBI, E.-G. **Metaheuristics: from Design to Implementation**. John Wiley & Sons, 2009.
- TANAKA, S.; FUJIKUMA, S. Na Efficient Exact Algorithm for General Single-Machine Scheduling with Machine Idle Time. **IEEE Conference on Automation Science and Engineering**, 2008.
- VALENTE, J. M. S.; MOREIRA, M. R. A. Greedy randomised dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. **International Journal of Advanced Technology**, v. 44, p. 995-1009, 2009.
- WAN, G.; YEN, B. P.-C. Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties. **European Journal of Operational Research**, v. 142, p. 271-281, 2002.
- WANG, P. Y.; WÄSCHER, G. Cutting and Packing. **European Journal of Operational Research**, v. 141, p. 239-240, 2002.
- WÄSCHER, G.; HAUßNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, p. 1109-1130, 2007.
- WIGHT, O. W. **Production and Inventory Management in the Computer Age**. New York: Van Nostrand Reinhold Company, Inc., 1984.
- YANASSE, H. H. On a pattern sequencing problem to minimize the maximum number of open stacks. **European Journal of Operational Research**, Amsterdam, v. 100, p. 454-463, 1997.
- YUEN, B. J. Heuristics for sequencing cutting patterns. **European Journal of Operational Research**, v. 55, p. 183-190, 1991.

APÊNDICE A – Código fonte para geração das instâncias em VBA

```
Option Explicit
```

```
Type rInst
```

```
    classe As Long  
    porte As Long  
    numRel As Long  
    numAbs As Long  
    w As Long  
    l As Long
```

```
    prazo As String  
    setup As String  
    custos As String
```

```
    st As Double  
    cBin As Double  
    m As Double
```

```
End Type
```

```
Public mInst() As rInst  
Public glQtdInst As Long
```

```
Type rItem
```

```
    numAbs As Long  
    it As Long  
    w As Long  
    l As Long  
    d As Double  
    a As Double  
    b As Double  
    p As Double
```

```
End Type
```

```
Public mItem() As rItem  
Public glQtdItem As Long
```

```
Sub criaInst()
```

```
Dim i As Long, j As Long, k As Long, f As Long, z As Long, m As Long  
Dim fAnt As Long, somaP As Double, maiorD As Double  
Dim appPath As String, arqPath As String, auxTxt As String
```

```
    Randomize  
    Application.DisplayAlerts = False
```

```
    ReDim mInst(1152)  
    ReDim mItem(69120)
```

```
    k = 1  
    f = 2  
    j = 1
```

```
    For i = 2 To 49  
        fAnt = f  
        For k = 1 To 24
```

```
            mInst(k + 24 * (i - 2)).classe = Cells(i, 1)  
            mInst(k + 24 * (i - 2)).porte = Cells(i, 2)  
            mInst(k + 24 * (i - 2)).numAbs = k + 24 * (i - 2)  
            mInst(k + 24 * (i - 2)).numRel = Cells(i, 4)  
            mInst(k + 24 * (i - 2)).l = Cells(i, 7)  
            mInst(k + 24 * (i - 2)).w = Cells(i, 8)
```

```
            f = fAnt  
            somaP = 0  
            maiorD = 0  
            Do while Cells(f, 13) = Cells(i, 3)
```



```

arqPath = mInst(k + 24 * (i - 2)).numAbs & ".dat"
Open appPath & arqPath For Output As 1
  auxTxt = "inst = " & mInst(k + 24 * (i - 2)).numAbs & _
  ";": Print #1, auxTxt
  auxTxt = "nItens = " & mInst(k + 24 * (i - 2)).porte & _
  ";": Print #1, auxTxt
  auxTxt = "cBin = " & mInst(k + 24 * (i - 2)).cBin & _
  ";": Print #1, auxTxt
  auxTxt = "L = " & mInst(k + 24 * (i - 2)).l & _
  ";": Print #1, auxTxt
  auxTxt = "w = " & mInst(k + 24 * (i - 2)).w & _
  ";": Print #1, auxTxt
  auxTxt = "st = " & mInst(k + 24 * (i - 2)).st & _
  ";": Print #1, auxTxt
  auxTxt = "M = " & mInst(k + 24 * (i - 2)).m & _
  ";": Print #1, auxTxt
  auxTxt = "alfa = _
  [ " & mItem(j - mInst(k + 24 * (i - 2)).porte).a
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).a
  Next
  auxTxt = auxTxt & " ];"
  Print #1, auxTxt

  auxTxt = "beta = [ " & _
  mItem(j - mInst(k + 24 * (i - 2)).porte).b
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).b
  Next
  auxTxt = auxTxt & " ];"
  Print #1, auxTxt

  auxTxt = "d = [ " & _
  mItem(j - mInst(k + 24 * (i - 2)).porte).d
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).d
  Next
  auxTxt = auxTxt & " ];"
  Print #1, auxTxt

  auxTxt = "p = [ " & _
  mItem(j - mInst(k + 24 * (i - 2)).porte).p
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).p
  Next
  auxTxt = auxTxt & " ];"
  Print #1, auxTxt

  auxTxt = "l = [ " & _
  mItem(j - mInst(k + 24 * (i - 2)).porte).l
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).l
  Next
  auxTxt = auxTxt & " ];"
  Print #1, auxTxt

  auxTxt = "w = [ " & _
  mItem(j - mInst(k + 24 * (i - 2)).porte).w
  For z = 2 To mInst(k + 24 * (i - 2)).porte
    auxTxt = auxTxt & ", " & _
    mItem(z - 1 + j - mInst(k + 24 * (i - 2)).porte).w
  Next
  auxTxt = auxTxt & " ];"

```

```

        Print #1, auxTxt
    Close 1
Next
Next
Application.DisplayAlerts = True

For i = 1 To UBound(mInst)
    worksheets("Inst").Cells(i + 1, 1) = mInst(i).classe
    worksheets("Inst").Cells(i + 1, 2) = mInst(i).porte
    worksheets("Inst").Cells(i + 1, 3) = mInst(i).numAbs
    worksheets("Inst").Cells(i + 1, 4) = mInst(i).numRel
    worksheets("Inst").Cells(i + 1, 5) = mInst(i).prazo
    worksheets("Inst").Cells(i + 1, 6) = mInst(i).setup
    worksheets("Inst").Cells(i + 1, 7) = mInst(i).custos
    worksheets("Inst").Cells(i + 1, 8) = mInst(i).cBin
    worksheets("Inst").Cells(i + 1, 9) = mInst(i).st
    worksheets("Inst").Cells(i + 1, 10) = mInst(i).l
    worksheets("Inst").Cells(i + 1, 11) = mInst(i).w
    worksheets("Inst").Cells(i + 1, 12) = mInst(i).m
Next

For i = 1 To UBound(mItem)
    worksheets("Inst").Cells(i + 1, 14) = mItem(i).numAbs
    worksheets("Inst").Cells(i + 1, 15) = mItem(i).it
    worksheets("Inst").Cells(i + 1, 16) = mItem(i).d
    worksheets("Inst").Cells(i + 1, 17) = mItem(i).a
    worksheets("Inst").Cells(i + 1, 18) = mItem(i).b
    worksheets("Inst").Cells(i + 1, 19) = mItem(i).p
    worksheets("Inst").Cells(i + 1, 20) = mItem(i).l
    worksheets("Inst").Cells(i + 1, 21) = mItem(i).w
Next

End Sub

```

APÊNDICE B – Código fonte para os experimentos no CPLEX através do IBM ILOG CPLEX Optimization Studio

B.1. Script (Script.mod)

```
main {  
  
    var source = new  
IloOplModelSource("C:/Users/hp/opl/Dissertação/Modelo.mod");  
    var cplex = new IloCplex();  
    var def = new IloOplModelDefinition(source);  
    var opl; var data; var before; var temp; var after;  
    var ofile = new IloOplOutputFile("Resultado.txt");  
  
    for (var k = 963; k <= 1152; k++)  
    {  
        opl = new IloOplModel(def, cplex);  
        data = new IloOplDataSource("C:/Users/hp/opl/Dissertação/" + k  
+ ".dat");  
        opl.addDataSource(data);  
        opl.generate();  
  
        before = new Date();  
        temp = before.getTime();  
  
        cplex.solve();  
        after = new Date();  
  
        ofile.writeln(k + " " +  
                        cplex.getObjValue() + " " +  
                        cplex.getBestObjValue() + " " +  
                        (after.getTime()-temp)/1000 + " " +  
                        opl.nBins + " " +  
                        opl.adiant + " " +  
                        opl.atraso );  
  
        data.end();  
        opl.end();  
    }  
  
    ofile.close();  
    def.end();  
    cplex.end();  
    source.end();  
}
```

B.2. Modelo (Modelo.mod)

```
execute PARAMS {
    cplex.tilim = 600;
}

// Índices e conjuntos
int nItens = ...; //Número de itens (n)
range Itens = 1..nItens; //Conjunto de itens
int inst = ...;

// Parâmetros
float cBin = ...; // Custo de um bin
float L = ...; // Comprimento do bin
float W = ...; // Largura do bin
float st = ...; //Tempo de preparação (setup) da máquina para receber um
lote
float alfa[Itens] = ...; // Custo por unidade de adiantamento do item i
float beta[Itens] = ...; // Custo por unidade de atraso do item i
float d[Itens] = ...; // Data de entrega do item i
float p[Itens] = ...; // Tempo de processamento do item i
float l[Itens] = ...; // Comprimento do item i
float w[Itens] = ...; // Largura do item i
float M = ...; // M grande

// Variáveis de decisão
dvar float+ s[Itens]; // Instante de término do item i
dvar float+ E[Itens]; // Adiantamento do item i
dvar float+ T[Itens]; // Atraso do item i
dvar float+ t[Itens]; // Tempo de processamento DO LOTE em que está o item
i
dvar float+ x[Itens]; // Coordenada x do canto inferior esquerdo do item i
dvar float+ y[Itens]; // Coordenada y do canto inferior esquerdo do item i
dvar float+ b[Itens]; // 1 se o item i "representa" um bin usado; 0 caso
contrário
dvar float+ adiant; // total de penalidades por adiantamento
dvar float+ atraso; // total de penalidades por atraso
dvar int+ nBins; // total de bins utilizados

dvar boolean X[Itens][Itens]; // 1, se o item i precede o item j no sentido
x; 0 caso contrário
dvar boolean Y[Itens][Itens]; // 1, se o item i precede o item j no sentido
y; 0 caso contrário
dvar boolean S[Itens][Itens]; // 1, se o item i precede o item j
temporalmente; 0 caso contrário

//Modelo:
minimize
    adiant + atraso + cBin * nBins;

subject to {

    forall( i in Itens ) T[i] >= s[i] + t[i] - d[i];
    forall( i in Itens ) E[i] >= d[i] - s[i] - t[i];
    forall( i in Itens ) x[i] <= L - l[i];
    forall( i in Itens ) y[i] <= W - w[i];
    forall( i in Itens ) t[i] == st + p[i] + sum( j in 1..(i-1) ) ( (1-
S[i][j]-S[j][i]) * (p[j]) ) +
```

```

sum( j in
(i+1)..nItems ) ( (1-S[i][j]-S[j][i]) * (p[j]) );

forall( i in Items ) b[i] >= 1 - sum(j in 1..(i-1)) (1 - S[i][j]-
S[j][i]);

forall( i in Items )
{
forall( j in 1..(i-1) ) s[i] + t[i] <= s[j] + M * (1 - S[i][j]);
forall( j in (i+1)..nItems ) s[i] + t[i] <= s[j] + M * (1 - S[i][j]);
}

forall( i in Items )
{
forall( j in 1..(i-1) ) s[i] <= s[j] + M * S[j][i];
forall( j in (i+1)..nItems ) s[i] <= s[j] + M * S[j][i];
}

forall( i in Items )
{
forall( j in 1..(i-1) ) x[i] - x[j] + L * ( X[i][j] - S[i][j] - S[j][i]
) <= L - l[i];
forall( j in (i+1)..nItems ) x[i] - x[j] + L * ( X[i][j] - S[i][j] -
S[j][i] ) <= L - l[i];
}

forall( i in Items )
{
forall( j in 1..(i-1) ) y[i] - y[j] + W * ( Y[i][j] - S[i][j] - S[j][i]
) <= W - w[i];
forall( j in (i+1)..nItems ) y[i] - y[j] + W * ( Y[i][j] - S[i][j] -
S[j][i] ) <= W - w[i];
}

forall( i in Items )
forall( j in 1..(i-1) ) X[i][j] + X[j][i] + Y[i][j] + Y[j][i] + S[i][j]
+ S[j][i] == 1;

nBins == sum(i in Items) (b[i]);
adiant == sum(i in Items) (alfa[i] * E[i]);
atraso == sum(i in Items) (beta[i] * T[i]);
}

```

B.3. Exemplo de arquivo de dados (I.dat)

```
inst = 1;
nItens = 20;
cBin = 10000;
L = 100;
W = 100;
st = 10.2;
M = 407.6;
alfa = [ 2.6, 4, 3, 1.3, 3.1, 1.4, 1.8, 2.7, 1.3, 3.7, 1.0, 1.4, 4.0, 1.0,
2.1, 2.1, 3.0, 4.5, 3.4, 2.4 ];
beta = [ 3.9, 4.2, 2.7, 2, 2.1, 4.1, 3.7, 1.5, 2.7, 4.5, 2.6, 4.3, 4.9,
3.2, 3.2, 4.4, 1.1, 2.6, 2.4, 5.0 ];
d = [ 5.1, 7.6, 13.6, 22.7, 27.1, 41.3, 48.2, 55.6, 64.9, 73.4, 84.1, 85.7,
83.6, 105.4, 110.4, 128.6, 141.2, 136, 143, 138.1 ];
p = [ 6.7, 4.5, 9.3, 13.7, 5.9, 19.9, 8.4, 2.6, 16.2, 14.2, 7.0, 9.8, 6.1,
17.3, 16.4, 7.4, 9.0, 3.8, 12.5, 12.8 ];
l = [ 75, 27, 24, 34, 1, 43, 6, 6, 38, 21, 49, 93, 37, 73, 4, 35, 48, 12,
79, 42 ];
w = [ 56, 98, 74, 99, 80, 84, 88, 89, 98, 78, 75, 31, 90, 21, 71, 93, 89,
90, 13, 10 ];
```

APÊNDICE C – Código fonte para a heurística construtiva em VBA

```
Sub geraUBinicialSolucao()

Dim i As Long, k As Long, j As Long
Dim min As Long, max As Long, tempoTotal As Double
Dim penalK As Double, penalKmenos As Double, penalKmais As Double
Dim passo As Long
Dim ant As Long, posAnt As Double, t As Double

ReDim mPlaca(qtdDadOrdInst)
ReDim mFaixa(qtdDadOrdInst)
ReDim mOrdem(qtdDadOrdInst)

qtdPlaca = 1: qtdFaixa = 1: qtdOrdem = 1

mPlaca(qtdPlaca).W = mDadInst.W
mPlaca(qtdPlaca).Lrest = mDadInst.L - mDadOrdInst(i).lj
mPlaca(qtdPlaca).faixaIni = 1
mPlaca(qtdPlaca).faixaFim = 1
mPlaca(qtdPlaca).qtdOrd = 1

mFaixa(qtdFaixa).Wrest = mDadInst.W - mDadOrdInst(i).wj
mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
mFaixa(qtdFaixa).X = 0

mOrdem(qtdOrdem).placa = qtdPlaca
mOrdem(qtdOrdem).xj = 0
mOrdem(qtdOrdem).yj = 0
mOrdem(qtdOrdem).j = i

For i = 2 To qtdDadOrdInst
    k = mPlaca(qtdPlaca).faixaIni
    Do while Not (mFaixa(k).Wrest >= mDadOrdInst(i).wj And _
mFaixa(k).L >= mDadOrdInst(i).lj)
        k = k + 1
        If k > mPlaca(qtdPlaca).faixaFim Then Exit Do
    Loop
    If k > mPlaca(qtdPlaca).faixaFim Then
        If mPlaca(qtdPlaca).W >= mDadOrdInst(i).wj And _
mPlaca(qtdPlaca).Lrest >= mDadOrdInst(i).lj Then

            'Restante da placa
            qtdFaixa = qtdFaixa + 1
            testeLimite "mFaixa"
            mPlaca(qtdPlaca).faixaFim = qtdFaixa
            mPlaca(qtdPlaca).Lrest = mPlaca(qtdPlaca).Lrest - _
mDadOrdInst(i).lj
            mPlaca(qtdPlaca).qtdOrd = mPlaca(qtdPlaca).qtdOrd + 1

            mFaixa(qtdFaixa).Wrest = mDadInst.W - mDadOrdInst(i).wj
            mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
            mFaixa(qtdFaixa).X = mDadInst.L - (mPlaca(qtdPlaca).Lrest _
+ mDadOrdInst(i).lj)

            qtdOrdem = qtdOrdem + 1
            testeLimite "mOrdem"
            mOrdem(qtdOrdem).placa = qtdPlaca
            mOrdem(qtdOrdem).xj = mDadInst.L - _
(mPlaca(qtdPlaca).Lrest + mDadOrdInst(i).lj)
            mOrdem(qtdOrdem).yj = 0
            mOrdem(qtdOrdem).j = i

        Else
```

```

        'Nova placa
        qtdPlaca = qtdPlaca + 1
        testeLimite "mPlaca"
        mPlaca(qtdPlaca).W = mDadInst.W
        mPlaca(qtdPlaca).Lrest = mDadInst.L - mDadOrdInst(i).lj
        mPlaca(qtdPlaca).faixaIni = qtdFaixa + 1
        mPlaca(qtdPlaca).faixaFim = qtdFaixa + 1
        mPlaca(qtdPlaca).qtdOrd = 1

        qtdFaixa = qtdFaixa + 1
        testeLimite "mFaixa"
        mFaixa(qtdFaixa).Wrest = mDadInst.W - mDadOrdInst(i).wj
        mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
        mFaixa(qtdFaixa).X = 0

        qtdOrdem = qtdOrdem + 1
        testeLimite "mOrdem"
        mOrdem(qtdOrdem).placa = qtdPlaca
        mOrdem(qtdOrdem).xj = 0
        mOrdem(qtdOrdem).yj = 0
        mOrdem(qtdOrdem).j = i
    End If
Else
    'Faixa existente
    qtdOrdem = qtdOrdem + 1
    testeLimite "mOrdem"
    mOrdem(qtdOrdem).placa = qtdPlaca
    mOrdem(qtdOrdem).xj = mFaixa(k).X
    mOrdem(qtdOrdem).yj = mDadInst.W - mFaixa(k).Wrest
    mOrdem(qtdOrdem).j = i

    mPlaca(qtdPlaca).qtdOrd = mPlaca(qtdPlaca).qtdOrd + 1
    mFaixa(k).Wrest = mFaixa(k).Wrest - mDadOrdInst(i).wj
End If

Next

UBsolu = mDadInst.cBin * qtdPlaca
posAnt = 0
QSmOrdem_placa_c 1, qtdOrdem
ant = 0
For i = 1 To qtdPlaca

    qtdPenal = mPlaca(i).qtdOrd
    ReDim auxPenal(qtdPenal)
    tempoTotal = mDadInst.setup

    For k = ant + 1 To ant + qtdPenal
        auxPenal(k - ant) = mOrdem(k).j
        tempoTotal = tempoTotal + mDadOrdInst(mOrdem(k).j).pj
    Next
    QS_auxPenal_dj_c 1, qtdPenal

    k = (qtdPenal + 1) \ 2
    min = 1
    max = qtdPenal
    If passo > 16 Then
        passo = qtdPenal \ 4
    Else
        passo = 1
    End If
    penalidades k, penalK, penalKmenos, penalKmais, passo

    Do Until (penalKmais - penalK) >= 0 And _
(penalK - penalKmenos) <= 0 And passo = 1
        If (penalKmais - penalK) < 0 Then
            min = k + passo
            k = (min + max) \ 2

```



```

        penalidades k, penalK, penalKmenos, penalKmais, passo
    ElseIf (penalK - penalKmenos) > 0 Then
        max = k - passo
        k = (min + max) \ 2
        penalidades k, penalK, penalKmenos, penalKmais, passo
    Else
        passo = passo \ 2
        penalidades k, penalK, penalKmenos, penalKmais, passo
    End If
Loop
If mDadOrdInst(auxPenal(k)).dj > (posAnt + tempoTotal) Then
    t = mDadOrdInst(auxPenal(k)).dj
Else
    t = posAnt + tempoTotal
End If
UBsolu = UBsolu + calc_Penalidade(t)

posAnt = t
ant = ant + qtdPenal
Next
End Sub

```

Apêndice D – Código fonte para a geração de *lower bound* segundo o método proposto em VBA

```

Option Explicit

Type rDadInst
    cBin As Double
    setup As Double
    L As Double
    W As Double
End Type
Public mDadInst As rDadInst

Type rDadOrdInst
    j As Long
    dj As Double
    aj As Double
    bj As Double
    pj As Double
    lj As Double
    wj As Double
End Type
Public mDadOrdInst() As rDadOrdInst, qtdDadOrdInst As Long

Type rSolucao
    proxSolu As Long
    nBins As Long
    soluPlacaIni As Long
    soluPlacaFim As Long
    LB As Long
End Type
Public mSolucao() As rSolucao, glQtdSolucao As Long
Public mSolucaoTemp() As rSolucao, qtdSoluTemp As Long
Public mSoluInsere As rSolucao
Public iniSoluTemp As Long

Type rSoluPlaca
    areaDisp As Double
    soluOrdIni As Long
    soluOrdFim As Long
End Type
Public mSoluPlaca() As rSoluPlaca, glQtdSoluPlaca As Long
Public mPlacaTemp() As rSoluPlaca, qtdPlacaTemp As Long
Public iniPlacaTemp As Long

Type rSoluOrd
    j As Long
    Cj As Double
End Type
Public mSoluOrd() As rSoluOrd, glQtdSoluOrd As Long
Public mOrdTemp() As rSoluOrd, qtdOrdTemp As Long
Public iniOrdTemp As Long

Public auxPenal() As Long
Public qtdPenal As Long

Public vInstaIni As Long, vInstaFim As Long
Public UBSolu As Double, LBbinPacking As Double
Public maxSol As Long, novoCj As Double, novoXj As Double, novoYj As Double
Public posResult As Long, tIni As Double, tLim As Double, tFim As Double
Public maxPlaca As Long, nItensInst As Long

Type rPlaca
    Lrest As Double
    W As Double

```

```

        qtdOrd As Long

        faixaIni As Long
        faixaFim As Long
    End Type
    Public mPlaca() As rPlaca
    Public qtdPlaca As Long

    Type rFaixa
        wrest As Double
        L As Double
        X As Double
    End Type
    Public mFaixa() As rFaixa
    Public qtdFaixa As Long

    Type rOrdem
        placa As Long
        xj As Double
        yj As Double
        j As Long
    End Type
    Public mOrdem() As rOrdem
    Public qtdOrdem As Long

    Sub ProgrCorte()

        Dim inst As Long, cod As Long, p As Long, ps As Long
        Dim s As Long, i As Long, o As Long, viavel As Boolean
        Dim incumb As Long

        'Otimização da rodada
        worksheets.Application.Calculation = xlCalculationManual
        worksheets.Application.ScreenUpdating = False

        'Set inicial de parâmetros
        posResult = 0
        AuxContaAndamento 2
        limpaResult
        tLim = worksheets("ProgCorte").Range("modDad").Cells(2, 3) / 86400

        'Leitura da faixa de instâncias
        vInstaIni = worksheets("ProgCorte").Range("modDad").Cells(2, 1)
        vInstaFim = worksheets("ProgCorte").Range("modDad").Cells(2, 2)

        For inst = vInstaIni To vInstaFim + 1 'Loop para cada instância

            CarregaDadosInst inst
            tIni = Now
            geraUBinicialSolucao
            geraLBbinPacking

            'Carrega dados da instância
            'Inicia contagem do tempo
            'Gerar UB inicial da solução
            'Gerar LB do bin packing associado

            'Encontra o máximo de placas a serem utilizadas
            maxPlaca = UBSolu \ mDadInst.cBin
            'Insere dados do primeiro item na primeira placa
            gera_solu_primeiro_item

            For i = 2 To qtdDadOrdInst 'Para cada item (i E I)

                'Zera temporárias
                ReDim mSolucaoTemp(5000): ReDim mPlacaTemp(10000)
                ReDim mOrdTemp(10000)
                qtdSoluTemp = 0: qtdPlacaTemp = 0: qtdOrdTemp = 0
                iniSoluTemp = 1: iniPlacaTemp = 1: iniOrdTemp = 1

                'Para cada solução parcial (s E S)
                For s = 1 To maxSol

                    'Gerar solução s' idêntica a s,
                    'colocando i em uma nova placa

```

```

        If mSolucao(s).nBins < maxPlaca Then
            ad_novaSolu True, s                'Nova solução
        'Copiar cada placa já existente
            copiar_placas_existentes i, s, -1
        'Se LBS' > UB então não incluir s' em S
define_viabilidade_LB                                End If

        'Para cada placa de s (p)
        For ps = mSolucao(s).soluPlacaIni To _
mSolucao(s).soluPlacaFim
            'Gerar solução s' idêntica a s, colocando i na placa p
            viavel = verifica_viabilidade_inserção(i, ps, s)
            If viavel Then
                ad_novaSolu False, s            'Nova solução
        'Copiar cada placa já existente
            copiar_placas_existentes i, s, ps
        'Se LBS' > UB então não incluir s' em S
            define_viabilidade_LB
            End If

            tFim = Now
        'Saída pelo tempo
            If (tFim - tIni) > tLim Then Exit For
        Next
        'Saída pelo tempo
        If (tFim - tIni) > tLim Then Exit For
    Next

    maxSol = qtdSoluTemp
    maxPlaca = (UBsolu - _
(mSolucaoTemp(1).LB - LBbinPacking * mDadInst.cBin)) \ _
mDadInst.cBin

    ReDim mSolucao(maxSol)
    mSolucao = mSolucaoTemp
    ReDim mSoluPlaca(qtdPlacaTemp)
    mSoluPlaca = mPlacaTemp
    ReDim mSoluOrd(qtdOrdTemp)
    mSoluOrd = mOrdTemp

    If (tFim - tIni) > tLim Then Exit For        'Saída pelo tempo
Next

    escreve_resultado inst, inst - vInstaIni + 1
    'Application.ThisWorkbook.Save

Next
AuxContaAndamento 3
'escreve_result_detalhes
worksheets.Application.Calculation = xlCalculationAutomatic
worksheets.Application.ScreenUpdating = True

End Sub

Sub CarregaDadosInst(inst As Long)

Dim i As Long, rg As Range, ant As Long

    'Leitura dos dados das instâncias
    Set rg = worksheets("Dados").Range("InstDadG1")
    i = 1
    Do while rg.Cells(i + 1, 1) <> "" And rg.Cells(i + 1, 1) <> inst
        i = i + 1
    Loop
    If rg.Cells(i + 1, 1) = "" Then
        Stop      'Instância não existe
    Else
        mDadInst.cBin = Round(rg.Cells(i + 1, 2), 1)
        mDadInst.setup = Round(rg.Cells(i + 1, 3), 1)
    End If
End Sub

```

```

        mDadInst.L = Round(rg.Cells(i + 1, 4), 1)
        mDadInst.W = Round(rg.Cells(i + 1, 5), 1)
    End If

    'Leitura dos dados das ordens das instâncias
    qtdDadOrdInst = 0
    Set rg = Worksheets("Dados").Range("OrdDadG1")
    i = 1
    Do While rg.Cells(i + 1, 1) <> "" And rg.Cells(i + 1, 1) <> inst
        i = i + 1
    Loop
    ant = i - 1
    ReDim mDadOrdInst(100)
    If rg.Cells(i + 1, 1) = "" Then
        Stop 'Instância não existe
    Else
        Do While rg.Cells(i + 1, 1) = inst
            qtdDadOrdInst = qtdDadOrdInst + 1
            mDadOrdInst(qtdDadOrdInst).j = rg.Cells(i + 1, 2)
            mDadOrdInst(qtdDadOrdInst).dj = Round(rg.Cells(i + 1, 3), 1)
            mDadOrdInst(qtdDadOrdInst).aj = Round(rg.Cells(i + 1, 4), 1)
            mDadOrdInst(qtdDadOrdInst).bj = Round(rg.Cells(i + 1, 5), 1)
            mDadOrdInst(qtdDadOrdInst).pj = Round(rg.Cells(i + 1, 6), 1)
            mDadOrdInst(qtdDadOrdInst).lj = Round(rg.Cells(i + 1, 7), 1)
            mDadOrdInst(qtdDadOrdInst).wj = Round(rg.Cells(i + 1, 8), 1)
            i = i + 1
        Loop
    End If
    ReDim Preserve mDadOrdInst(qtdDadOrdInst)
    QSmDadOrdInst_dj_c 1, qtdDadOrdInst

End Sub

Sub geraLBbinPacking()

Dim i As Long, div As Long, cont As Long
Dim areaRestPlaca As Double, areaRestOrd As Double

    cont = 0: areaRestPlaca = 0: areaRestOrd = 0
    LBbinPacking = 0
    For i = 1 To qtdDadOrdInst
        LBbinPacking = LBbinPacking + mDadOrdInst(i).wj * mDadOrdInst(i).lj
        If mDadOrdInst(i).wj > (mDadInst.W / 2) And _
mDadOrdInst(i).lj > (mDadInst.L / 2) Then
            areaRestPlaca = areaRestPlaca + mDadInst.W * mDadInst.L - _
                mDadOrdInst(i).wj * mDadOrdInst(i).lj
            cont = cont + 1
        Else
            areaRestOrd = areaRestOrd + _
mDadOrdInst(i).wj * mDadOrdInst(i).lj
        End If
    Next

    'LB pela área
    LBbinPacking = LBbinPacking / (mDadInst.L * mDadInst.W)
    div = Round(LBbinPacking, 0)
    If div > LBbinPacking Then div = div - 1

    'LB pelo número de itens
    If areaRestOrd > areaRestPlaca Then
        div = Round((areaRestOrd - areaRestPlaca) / _
(mDadInst.L * mDadInst.W), 0)
        If div > (areaRestOrd - areaRestPlaca) / (mDadInst.L * mDadInst.W)_
Then div = div - 1
    Else
        div = 0
    End If
    cont = cont + div

```

```

    If LBbinPacking < cont Then
        LBbinPacking = cont
    End If
End Sub

Sub geraUBinicialSolucao()

Dim i As Long, k As Long, j As Long
Dim min As Long, max As Long, tempoTotal As Double
Dim penalK As Double, penalKmenos As Double, penalKmais As Double
Dim passo As Long
Dim ant As Long, posAnt As Double, t As Double

    ReDim mPlaca(qtdDadOrdInst)
    ReDim mFaixa(qtdDadOrdInst)
    ReDim mOrdem(qtdDadOrdInst)

    qtdPlaca = 1: qtdFaixa = 1: qtdOrdem = 1

    mPlaca(qtdPlaca).W = mDadInst.W
    mPlaca(qtdPlaca).Lrest = mDadInst.L - mDadOrdInst(i).lj
    mPlaca(qtdPlaca).faixaIni = 1
    mPlaca(qtdPlaca).faixaFim = 1
    mPlaca(qtdPlaca).qtdOrd = 1

    mFaixa(qtdFaixa).Wrest = mDadInst.W - mDadOrdInst(i).wj
    mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
    mFaixa(qtdFaixa).X = 0

    mOrdem(qtdOrdem).placa = qtdPlaca
    mOrdem(qtdOrdem).xj = 0
    mOrdem(qtdOrdem).yj = 0
    mOrdem(qtdOrdem).j = i

    For i = 2 To qtdDadOrdInst

        k = mPlaca(qtdPlaca).faixaIni
        Do While Not (mFaixa(k).Wrest >= mDadOrdInst(i).wj And _
mFaixa(k).L >= mDadOrdInst(i).lj)
            k = k + 1
            If k > mPlaca(qtdPlaca).faixaFim Then Exit Do
        Loop
        If k > mPlaca(qtdPlaca).faixaFim Then
            If mPlaca(qtdPlaca).W >= mDadOrdInst(i).wj And _
mPlaca(qtdPlaca).Lrest >= mDadOrdInst(i).lj Then

                'Restante da placa
                qtdFaixa = qtdFaixa + 1
                testeLimite "mFaixa"
                mPlaca(qtdPlaca).faixaFim = qtdFaixa
                mPlaca(qtdPlaca).Lrest = mPlaca(qtdPlaca).Lrest - _
                    mDadOrdInst(i).lj
                mPlaca(qtdPlaca).qtdOrd = mPlaca(qtdPlaca).qtdOrd + 1

                mFaixa(qtdFaixa).Wrest = mDadInst.W - mDadOrdInst(i).wj
                mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
                mFaixa(qtdFaixa).X = mDadInst.L - (mPlaca(qtdPlaca).Lrest _
+ mDadOrdInst(i).lj)

                qtdOrdem = qtdOrdem + 1
                testeLimite "mOrdem"
                mOrdem(qtdOrdem).placa = qtdPlaca
                mOrdem(qtdOrdem).xj = mDadInst.L - _
(mPlaca(qtdPlaca).Lrest + mDadOrdInst(i).lj)
                mOrdem(qtdOrdem).yj = 0
                mOrdem(qtdOrdem).j = i
            End If
        End If
    Next i
End Sub

```

```

Else
    'Nova placa
    qtdPlaca = qtdPlaca + 1
    testeLimite "mPlaca"
    mPlaca(qtdPlaca).W = mDadInst.W
    mPlaca(qtdPlaca).Lrest = mDadInst.L - mDadOrdInst(i).lj
    mPlaca(qtdPlaca).faixaIni = qtdFaixa + 1
    mPlaca(qtdPlaca).faixaFim = qtdFaixa + 1
    mPlaca(qtdPlaca).qtdOrd = 1

    qtdFaixa = qtdFaixa + 1
    testeLimite "mFaixa"
    mFaixa(qtdFaixa).wrest = mDadInst.W - mDadOrdInst(i).wj
    mFaixa(qtdFaixa).L = mDadOrdInst(i).lj
    mFaixa(qtdFaixa).X = 0

    qtdOrdem = qtdOrdem + 1
    testeLimite "mOrdem"
    mOrdem(qtdOrdem).placa = qtdPlaca
    mOrdem(qtdOrdem).xj = 0
    mOrdem(qtdOrdem).yj = 0
    mOrdem(qtdOrdem).j = i

End If
Else
    'Faixa existente
    qtdOrdem = qtdOrdem + 1
    testeLimite "mOrdem"
    mOrdem(qtdOrdem).placa = qtdPlaca
    mOrdem(qtdOrdem).xj = mFaixa(k).X
    mOrdem(qtdOrdem).yj = mDadInst.W - mFaixa(k).wrest
    mOrdem(qtdOrdem).j = i

    mPlaca(qtdPlaca).qtdOrd = mPlaca(qtdPlaca).qtdOrd + 1
    mFaixa(k).wrest = mFaixa(k).wrest - mDadOrdInst(i).wj
End If

Next

UBsolu = mDadInst.cBin * qtdPlaca
posAnt = 0
QSmOrdem_placa_c 1, qtdOrdem
ant = 0
For i = 1 To qtdPlaca

    qtdPenal = mPlaca(i).qtdOrd
    ReDim auxPenal(qtdPenal)
    tempoTotal = mDadInst.setup

    For k = ant + 1 To ant + qtdPenal
        auxPenal(k - ant) = mOrdem(k).j
        tempoTotal = tempoTotal + mDadOrdInst(mOrdem(k).j).pj
    Next
    QS_auxPenal_dj_c 1, qtdPenal

    k = (qtdPenal + 1) \ 2
    min = 1
    max = qtdPenal
    If passo > 16 Then
        passo = qtdPenal \ 4
    Else
        passo = 1
    End If
    penalidades k, penalK, penalKmenos, penalKmais, passo

    Do Until (penalKmais - penalK) >= 0 And _
(penalK - penalKmenos) <= 0 And passo = 1
        If (penalKmais - penalK) < 0 Then
            min = k + passo

```

```

        k = (min + max) \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    ElseIf (penalk - penalkmenos) > 0 Then
        max = k - passo
        k = (min + max) \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    Else
        passo = passo \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    End If
Loop

If mDadOrdInst(auxPenal(k)).dj > (posAnt + tempoTotal) Then
    t = mDadOrdInst(auxPenal(k)).dj
Else
    t = posAnt + tempoTotal
End If
UBsolu = UBSolu + calc_Penalidade(t)

posAnt = t
ant = ant + qtdPenal
Next

End Sub

Function verifica_viabilidade_inserção(ByVal i As Long, _
    ByVal ps As Long, ByVal s As Long) As Boolean

    If mSoluPlaca(ps).areaDisp < (mDadOrdInst(i).lj * mDadOrdInst(i).wj) _
        Then
        verifica_viabilidade_inserção = False
    Else
        verifica_viabilidade_inserção = True
        novoCj = melhorPos(ps, i)
    End If

End Function

Function melhorPos(ByVal ps As Long, ByVal i As Long) As Double

Dim k As Long, min As Long, max As Long, tempoTotal As Double
Dim penalk As Double, penalkmenos As Double, penalkmais As Double
Dim passo As Long

tempoTotal = mDadInst.setup

qtdPenal = mSoluPlaca(ps).soluOrdFim - mSoluPlaca(ps).soluOrdIni + 2
ReDim auxPenal(qtdPenal)
For k = mSoluPlaca(ps).soluOrdIni To mSoluPlaca(ps).soluOrdFim
    auxPenal(k - mSoluPlaca(ps).soluOrdIni + 1) = mSoluOrd(k).j
    tempoTotal = tempoTotal + mDadOrdInst(mSoluOrd(k).j).pj
Next
auxPenal(qtdPenal) = i
tempoTotal = tempoTotal + mDadOrdInst(i).pj

k = (qtdPenal + 1) \ 2
min = 1
max = qtdPenal
If passo > 16 Then
    passo = qtdPenal \ 4
Else
    passo = 1
End If
penalidades k, penalk, penalkmenos, penalkmais, passo

Do Until (penalkmais - penalk) >= 0 And (penalk - penalkmenos) <= 0 _
And passo = 1
    If (penalkmais - penalk) < 0 Then
        min = k + passo
    
```



```

        k = (min + max) \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    ElseIf (penalk - penalkmenos) > 0 Then
        max = k - passo
        k = (min + max) \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    Else
        passo = passo \ 2
        penalidades k, penalk, penalkmenos, penalkmais, passo
    End If
Loop

If mDadOrdInst(auxPenal(k)).dj > tempoTotal Then
    melhorPos = mDadOrdInst(auxPenal(k)).dj
Else
    melhorPos = tempoTotal
End If

End Function

Function calc_Penalidade(ByVal t As Double) As Double

Dim k As Long

    calc_Penalidade = 0
    For k = 1 To qtdPenal
        If mDadOrdInst(auxPenal(k)).dj < t Then
            calc_Penalidade = calc_Penalidade + _
(t - mDadOrdInst(auxPenal(k)).dj) * mDadOrdInst(auxPenal(k)).bj
        Else
            calc_Penalidade = calc_Penalidade + _
(mDadOrdInst(auxPenal(k)).dj - t) * mDadOrdInst(auxPenal(k)).aj
        End If
    Next

End Function

Sub limpaResult()

Dim rg As Range

    Set rg = worksheets("ProgCorte").Range("InstResult")
    If rg.Cells(2, 1) <> "" Then
        Range(rg.Cells(2, 1), _
rg.Cells(2,1).End(xlToRight).End(xlDown)).ClearContents
    End If

    Set rg = worksheets("ProgCorte").Range("OrdResult")
    If rg.Cells(2, 1) <> "" Then
        Range(rg.Cells(2, 1), _
rg.Cells(2,1).End(xlToRight).End(xlDown)).ClearContents
    End If

End Sub

Sub escreve_resultado(ByVal inst As Long, ByVal ord As Long)

Dim rg As Range

    Set rg = worksheets("ProgCorte").Range("InstResult")
    rg.Cells(ord + 1, 1) = inst
    rg.Cells(ord + 1, 2) = UBSolu
    rg.Cells(ord + 1, 3) = (tFim - tIni) * 86400
    rg.Cells(ord + 1, 4) = mSolucao(1).nBins
    rg.Cells(ord + 1, 5) = mSolucao(1).LB
    rg.Cells(ord + 1, 6) = maxSol

End Sub

```

```

Sub penalidades(ByVal k As Long, ByRef penalK As Double, ByRef penalKmenos
As Double, ByRef penalKmais As Double, ByVal passo As Long)
    penalK = calc_Penalidade(mDadOrdInst(auxPenal(k)).dj)
    If k > 1 Then
        penalKmenos = calc_Penalidade(mDadOrdInst(auxPenal(k - passo)).dj)
    Else
        penalKmenos = penalK + 1
    End If
    If k < qtdPenal Then
        penalKmais = calc_Penalidade(mDadOrdInst(auxPenal(k + passo)).dj)
    Else
        penalKmais = penalK + 1
    End If
End Sub

Sub gera_solu_primeiro_item()
    ReDim mSolucao(1)
    ReDim mSoluPlaca(1)
    ReDim mSoluOrd(1)

    maxSol = 1
    glQtdSolucao = 1
    glQtdSoluPlaca = 1
    glQtdSoluOrd = 1

    mSolucao(1).proxSolu = -1
    mSolucao(1).LB = mDadInst.cBin * LBbinPacking
    mSolucao(1).nBins = 1
    mSolucao(1).soluPlacaIni = 1
    mSolucao(1).soluPlacaFim = 1

    mSoluPlaca(1).areaDisp = mDadInst.L * _
mDadInst.W - mDadOrdInst(1).wj * mDadOrdInst(1).lj
    mSoluPlaca(1).soluOrdIni = 1
    mSoluPlaca(1).soluOrdFim = 1

    mSoluOrd(1).j = 1
    If mDadOrdInst(1).dj > mDadOrdInst(1).pj + mDadInst.setup Then
        mSoluOrd(1).Cj = mDadOrdInst(1).dj
    Else
        mSoluOrd(1).Cj = mDadOrdInst(1).pj + mDadInst.setup
        mSolucao(1).LB = mSolucao(1).LB + _
(mSoluOrd(1).Cj - mDadOrdInst(1).dj) * mDadOrdInst(1).bj
    End If
End Sub

Sub define_viabilidade_LB()
    Dim p As Long, o As Long, s As Long
    Dim sAnt As Long

    'Se LBs' > UB então não incluir s' em S
    If mSoluInsere.LB <= UBsolu Then

        'Incrementa índice
        qtdSoluTemp = qtdSoluTemp + 1
        testeLimite "mSolucaoTemp"

        'Amarra índices
        mSolucaoTemp(qtdSoluTemp) = mSoluInsere

        'Posiciona na lista
        If qtdSoluTemp = 1 Then
            mSolucaoTemp(qtdSoluTemp).proxSolu = -1
        Else

```

```

        s = iniSoluTemp
        sAnt = -1
        Do while mSolucaoTemp(qtdSoluTemp).LB > mSolucaoTemp(s).LB
            If mSolucaoTemp(s).proxSolu = -1 Then Exit Do
            sAnt = s
            s = mSolucaoTemp(s).proxSolu
        Loop

        If mSolucaoTemp(s).proxSolu = -1 Then
            mSolucaoTemp(s).proxSolu = qtdSoluTemp
            mSolucaoTemp(qtdSoluTemp).proxSolu = -1
        Else
            If sAnt = -1 Then
                iniSoluTemp = qtdSoluTemp
                mSolucaoTemp(qtdSoluTemp).proxSolu = s
            Else
                mSolucaoTemp(sAnt).proxSolu = qtdSoluTemp
                mSolucaoTemp(qtdSoluTemp).proxSolu = s
            End If
        End If
    End If

End If

End Sub

Sub testeLimite(ByVal matriz As String)
    Select Case matriz
        Case "mSoluPlaca"
            If glQtdSoluPlaca > UBound(mSoluPlaca) Then
                ReDim Preserve mSoluPlaca(UBound(mSoluPlaca) + 10000)
            End If
        Case "mSoluOrd"
            If glQtdSoluOrd > UBound(mSoluOrd) Then
                ReDim Preserve mSoluOrd(UBound(mSoluOrd) + 100000)
            End If
        Case "mSolucao"
            If glQtdSolucao > UBound(mSolucao) Then
                ReDim Preserve mSolucao(UBound(mSolucao) + 1000)
            End If
        Case "mFaixa"
            If qtdFaixa > UBound(mFaixa) Then
                ReDim Preserve mFaixa(qtdFaixa + 100)
            End If
        Case "mOrdem"
            If qtdOrdem > UBound(mOrdem) Then
                ReDim Preserve mOrdem(qtdOrdem + 100)
            End If
        Case "mPlaca"
            If qtdPlaca > UBound(mPlaca) Then
                ReDim Preserve mPlaca(qtdPlaca + 10)
            End If
        Case "mPlacaTemp"
            If qtdPlacaTemp > UBound(mPlacaTemp) Then
                ReDim Preserve mPlacaTemp(UBound(mPlacaTemp) + 10000)
            End If
        Case "mOrdTemp"
            If qtdOrdTemp > UBound(mOrdTemp) Then
                ReDim Preserve mOrdTemp(UBound(mOrdTemp) + 100000)
            End If
        Case "mSolucaoTemp"
            If qtdSoluTemp > UBound(mSolucaoTemp) Then
                ReDim Preserve mSolucaoTemp(UBound(mSolucaoTemp) + 5000)
            End If
    End Select
End Sub

```

```

Sub ad_nova_Placa(ByVal i As Long)
    'Gera placa nova
    qtdPlacaTemp = qtdPlacaTemp + 1
    testeLimite "mPlacaTemp"
    mSoluInserere.soluPlacaFim = qtdPlacaTemp
    mPlacaTemp(qtdPlacaTemp).areaDisp = mDadInst.L * _
mDadInst.W - mDadOrdInst(i).wj * mDadOrdInst(i).lj

    'Inserere item novo (i) na placa nova
    qtdOrdTemp = qtdOrdTemp + 1
    testeLimite "mOrdTemp"

    mPlacaTemp(qtdPlacaTemp).soluOrdIni = qtdOrdTemp
    mPlacaTemp(qtdPlacaTemp).soluOrdFim = qtdOrdTemp

    mOrdTemp(qtdOrdTemp).j = i

    'Define Cj e já atualiza o LB
    If mDadOrdInst(i).dj > mDadOrdInst(i).pj + mDadInst.setup Then
        mOrdTemp(qtdOrdTemp).Cj = mDadOrdInst(i).dj
    Else
        mOrdTemp(qtdOrdTemp).Cj = mDadOrdInst(i).pj + mDadInst.setup
        mSoluInserere.LB = mSoluInserere.LB + (mOrdTemp(qtdOrdTemp).Cj - _
mDadOrdInst(i).dj) * mDadOrdInst(i).bj
    End If
End Sub

Sub ad_novaSolu(novo As Boolean, ByVal s As Long)
    'Número de bins e LB por material
    If novo Then
        mSoluInserere.nBins = mSolucao(s).nBins + 1
        If mSoluInserere.nBins > LBbinPacking Then
            mSoluInserere.LB = mSolucao(s).LB + mDadInst.cBin
        Else
            mSoluInserere.LB = mSolucao(s).LB
        End If
    Else
        mSoluInserere.nBins = mSolucao(s).nBins
        mSoluInserere.LB = mSolucao(s).LB
    End If
End Sub

Sub copiar_placas_existentes(ByVal i As Long, ByVal s As Long, _
ByVal ps As Long)
    Dim p As Long, o As Long, aux As Long

    mSoluInserere.soluPlacaIni = glQtdSoluPlaca + 1
    For p = mSolucao(s).soluPlacaIni To mSolucao(s).soluPlacaFim

        qtdPlacaTemp = qtdPlacaTemp + 1
        testeLimite "mPlacaTemp"

        If p = ps Then
            mPlacaTemp(qtdPlacaTemp).areaDisp = _
mPlacaTemp(p).areaDisp - mDadOrdInst(i).lj * mDadOrdInst(i).wj
        Else
            mPlacaTemp(qtdPlacaTemp).areaDisp = mPlacaTemp(p).areaDisp
        End If

        mPlacaTemp(qtdPlacaTemp).soluOrdIni = qtdOrdTemp + 1
        For o = mSoluPlaca(p).soluOrdIni To mSoluPlaca(p).soluOrdFim
            qtdOrdTemp = qtdOrdTemp + 1
            testeLimite "mOrdTemp"
        Next o
    Next p
End Sub

```

```

mOrdTemp(qtdOrdTemp).j = mSoluOrd(o).j

If p = ps Then
    aux = mSoluOrd(o).j
    If mDadOrdInst(aux).dj > mSoluOrd(o).Cj And _
        mDadOrdInst(aux).dj > novocj Then
        mSoluInsere.LB = mSoluInsere.LB _
        - (mDadOrdInst(aux).dj - mSoluOrd(o).Cj) * mDadOrdInst(aux).aj _
        + (mDadOrdInst(aux).dj - novocj) * mDadOrdInst(aux).aj

        ElseIf mDadOrdInst(aux).dj > mSoluOrd(o).Cj And _
            mDadOrdInst(aux).dj < novocj Then
            mSoluInsere.LB = mSoluInsere.LB _
            - (mDadOrdInst(aux).dj - mSoluOrd(o).Cj) * mDadOrdInst(aux).aj _
            + (novocj - mDadOrdInst(aux).dj) * mDadOrdInst(aux).bj

        ElseIf mDadOrdInst(aux).dj < mSoluOrd(o).Cj And _
            mDadOrdInst(aux).dj > novocj Then
            mSoluInsere.LB = mSoluInsere.LB _
            - (mSoluOrd(o).Cj - mDadOrdInst(aux).dj) * mDadOrdInst(aux).bj _
            + (mDadOrdInst(aux).dj - novocj) * mDadOrdInst(aux).aj

        Else
            mSoluInsere.LB = mSoluInsere.LB _
            - (mSoluOrd(o).Cj - mDadOrdInst(aux).dj) * mDadOrdInst(aux).bj _
            + (novocj - mDadOrdInst(aux).dj) * mDadOrdInst(aux).bj
        End If
        mOrdTemp(qtdOrdTemp).Cj = novocj
    Else
        mOrdTemp(qtdOrdTemp).Cj = mSoluOrd(o).Cj
    End If
Next

If p = ps Then insere_novo_item i
mPlacaTemp(qtdPlacaTemp).soluOrdFim = qtdOrdTemp
Next

If ps = -1 Then
    ad_nova_Placa i 'Colocar a nova placa
Else
    mSoluInsere.soluPlacaFim = qtdPlacaTemp
End If

End Sub

Sub insere_novo_item(ByVal i As Long)

    qtdOrdTemp = qtdOrdTemp + 1
    testeLimite "mOrdTemp"

    mOrdTemp(qtdOrdTemp).j = i
    mOrdTemp(qtdOrdTemp).Cj = novocj

    If mDadOrdInst(i).dj > novocj Then
        mSoluInsere.LB = mSoluInsere.LB _
        + (mDadOrdInst(i).dj - novocj) * mDadOrdInst(i).aj
    Else
        mSoluInsere.LB = mSoluInsere.LB _
        + (novocj - mDadOrdInst(i).dj) * mDadOrdInst(i).bj
    End If

End Sub

End Sub

```

```

Sub QSmDadOrdInst_dj_c(iMin As Long, Imax As Long)

Dim tReg As rDadOrdInst
Dim j As Long, k As Long
Dim Meio As Double, Lmin As Long, LMax As Long

    Lmin = iMin
    LMax = Imax
    k = iMin
    j = Imax
    Meio = mDadOrdInst((Lmin + LMax) \ 2).dj
    Do
        Do while mDadOrdInst(k).dj < Meio
            k = k + 1
        Loop
        Do while Meio < mDadOrdInst(j).dj
            j = j - 1
        Loop
        If k < j Then
            tReg = mDadOrdInst(k)
            mDadOrdInst(k) = mDadOrdInst(j)
            mDadOrdInst(j) = tReg
        End If
        If k <= j Then
            k = k + 1
            j = j - 1
        End If
    Loop Until k > j
    If iMin < j Then QSmDadOrdInst_dj_c iMin, j
    If k < Imax Then QSmDadOrdInst_dj_c k, Imax
End Sub

Sub AuxContaAndamento(ByVal i As Long)

Dim rg As Range, aux As Double

    Set rg = Worksheets("ProgCorte").Range("And")
    aux = Now
    rg.Cells(i, 2) = Format$(aux, "hh mm ss")

End Sub

Sub QS_auxPenal_dj_c(iMin As Long, Imax As Long)

Dim tReg As Long
Dim j As Long, k As Long
Dim Meio As Double, Lmin As Long, LMax As Long

    Lmin = iMin
    LMax = Imax
    k = iMin
    j = Imax
    Meio = mDadOrdInst(auxPenal((Lmin + LMax) \ 2)).dj
    Do
        Do while mDadOrdInst(auxPenal(k)).dj < Meio
            k = k + 1
        Loop
        Do while Meio < mDadOrdInst(auxPenal(j)).dj
            j = j - 1
        Loop
        If k < j Then
            tReg = auxPenal(k)
            auxPenal(k) = auxPenal(j)
            auxPenal(j) = tReg
        End If
        If k <= j Then
            k = k + 1
            j = j - 1
        End If
    Loop Until k > j
    If iMin < j Then QS_auxPenal_dj_c iMin, j
    If k < Imax Then QS_auxPenal_dj_c k, Imax
End Sub

```

```

        End If
    Loop Until k > j
    If iMin < j Then QS_auxPenal_dj_c iMin, j
    If k < Imax Then QS_auxPenal_dj_c k, Imax
End Sub

Sub QSmOrdem_placa_c(iMin As Long, Imax As Long)

Dim tReg As rOrdem
Dim j As Long, k As Long
Dim Meio As Long, Lmin As Long, LMax As Long

    Lmin = iMin
    LMax = Imax
    k = iMin
    j = Imax
    Meio = mOrdem((Lmin + LMax) \ 2).placa
    Do
        Do While mOrdem(k).placa < Meio
            k = k + 1
        Loop
        Do While Meio < mOrdem(j).placa
            j = j - 1
        Loop
        If k < j Then
            tReg = mOrdem(k)
            mOrdem(k) = mOrdem(j)
            mOrdem(j) = tReg
        End If
        If k <= j Then
            k = k + 1
            j = j - 1
        End If
    Loop Until k > j
    If iMin < j Then QSmOrdem_placa_c iMin, j
    If k < Imax Then QSmOrdem_placa_c k, Imax
End Sub

```

APÊNDICE E – Detalhamento de testes estatísticos

Tabela E-1 – Sumário de estatísticas descritivas para a ANOVA dos *gaps* obtidos pelo CPLEX entre as classes de tamanhos de itens

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Classe 7	288	1743531,4%	6053,9%	472886,0%
Classe 8	288	1919741,4%	6665,8%	517500,7%
Classe 9	288	835625,1%	2901,5%	207666,7%
Classe 10	288	1933018,9%	6711,9%	585666,1%

Tabela E-2 – ANOVA dos *gaps* obtidos pelo CPLEX entre as classes de tamanhos de itens

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	283942	3	94647	21,2247	< 0,0001	2,6127
Nos grupos	5119275	1148	4459			
Total	5403216	1151				

Tabela E-3 – Sumário de estatísticas descritivas para a ANOVA dos *gaps* obtidos pelos métodos propostos entre as classes de tamanhos de itens

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Classe 7	288	271477,7%	942,6%	18455,4%
Classe 8	288	323721,5%	1124,0%	26376,4%
Classe 9	288	164128,3%	569,9%	8044,8%
Classe 10	288	331168,8%	1149,9%	28356,9%

Tabela E-4 – ANOVA dos *gaps* obtidos pelos métodos propostos entre as classes de tamanhos de itens

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	6184	3	2061	10,1508	< 0,0001	2,6127
Nos grupos	233140	1148	203			
Total	239324	1151				

Tabela E-5 – Teste t de Student de diferença de médias dos *upper bounds* obtidos pelo CPLEX e pela heurística construtiva proposta com 5% de significância

	Limitantes superiores	
	CPLEX	Heurística construtiva
Média	4347570,8	2500375,4
Variância	5,68909E+13	1,9634E+13
Observações	1152	1152
Graus de liberdade	1861	
t de Student	7,1670	
valor p	< 0,0001	
t crítico	1,9612	

Tabela E-6 – Teste t de Student de diferença de médias dos *lower bounds* obtidos pelo CPLEX e pela heurística construtiva proposta com 5% de significância

	Limitantes inferiores	
	CPLEX	Método proposto
Média	102262,4	211456,1
Variância	14919414026	31900620547
Observações	1152	1152
Graus de liberdade	2034	
t de Student	-17,1281	
valor p	< 0,0001	
t crítico	1,9611	

Tabela E-7 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *upper bounds* da heurística construtiva e do CPLEX entre as classes de tamanhos de itens

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Classe 7	288	-5663,1%	-19,7%	24,2%
Classe 8	288	-2537,1%	-8,8%	28,3%
Classe 9	288	-131,2%	-0,5%	6,4%
Classe 10	288	-6420,9%	-22,3%	32,6%

Tabela E-8 – ANOVA das comparações percentuais dos *upper bounds* da heurística construtiva e do CPLEX entre as classes de tamanhos de itens

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	8,8003	3	2,9334	12,8212	< 0,0001	2,6127
Nos grupos	262,6583	1148	0,2288			
Total	271,4586	1151				

Tabela E-9 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre as classes de tamanhos de itens

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Classe 7	288	114294,3%	396,9%	2399,6%
Classe 8	288	138384,8%	480,5%	3192,6%
Classe 9	288	117985,1%	409,7%	4774,0%
Classe 10	288	78734,6%	273,4%	1178,8%

Tabela E-10 – ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre as classes de tamanhos de itens

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	640,0	3	213,3496	7,3919	< 0,0001	2,6127
Nos grupos	33134,1	1148	28,8625			
Total	33774,2	1151				

Tabela E-11 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre os portes de instâncias (número de itens)

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
20 itens	384	18285,9%	47,6%	6,5%
60 itens	384	-14738,5%	-38,4%	5,6%
100 itens	384	-18299,7%	-47,7%	3,4%

Tabela E-12 – ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre os portes de instâncias (número de itens)

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	211,96	2	105,9807	2046,6807	< 0,0001	3,0036
Nos grupos	59,50	1149	0,051782			
Total	271,46	1151				

Tabela E-13 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre os portes de instâncias (número de itens)

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
20 itens	384	31249,1%	81,4%	148,0%
60 itens	384	132076,5%	343,9%	1104,7%
100 itens	384	286073,2%	745,0%	5326,0%

Tabela E-14 – ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre os portes de instâncias (número de itens)

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	8577,82	2	4288,9120	195,5821	< 0,0001	3,0036
Nos grupos	25196,37	1149	21,9290			
Total	33774,19	1151				

Tabela E-15 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
α e β baixos	288	-3983,5%	-13,8%	22,6%
α alto e β baixo	288	-5688,9%	-19,8%	25,4%
α baixo e β alto	288	-2158,8%	-7,5%	22,4%
α e β altos	288	-2921,2%	-10,1%	23,4%

Tabela E-16 – ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	2,43656	3	0,8122	3,4659	0,01578	2,6127
Nos grupos	269,0220	1148	0,2343			
Total	271,4586	1151				

Tabela E-17 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
α e β baixos	288	171975,2%	597,1%	3521,9%
α alto e β baixo	288	174571,9%	606,2%	3956,4%
α baixo e β alto	288	52541,6%	182,4%	1238,9%
α e β altos	288	50310,1%	174,7%	1253,8%

Tabela E-18 – ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre os custos unitários de adiantamento (α) e atraso (β)

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	5157,20	3	1719,0650	68,9620	< 0,0001	2,6127
Nos grupos	28617,00	1148	24,927699			
Total	33774,19	1151				

Tabela E- 19 – Teste t de Student de diferença de médias da comparação percentual dos *lower bounds* para instâncias de custos de adiantamento (α) altos e baixos com 5% de significância

	Custo de adiantamento (α)	
	<i>Alto</i>	<i>Baixo</i>
Média	390,4%	389,8%
Variância	3066,8%	2807,0%
Observações	576	576
Graus de liberdade	1148	
t	0,0199	
valor-p	0,9842	
t crítico	1,9620	

Tabela E-20 – Teste t de Student de diferença de médias da comparação percentual dos *upper bounds* para instâncias de *setups* curtos e longos com 5% de significância

	Setup	
	Curto	Longo
Média	-15,26%	-10,35%
Variância	18,16%	28,93%
Observações	576	576
Graus de liberdade	1093	
t de Student	-1,7182	
valor p	0,0860	
t crítico	1,9621	

Tabela E-21 – Teste t de Student de diferença de médias da comparação percentual dos *lower bounds* para instâncias de *setups* curtos e longos com 5% de significância

	Setup	
	Curto	Longo
Média	567,4%	212,8%
Variância	3805,5%	1438,6%
Observações	576	576
Graus de liberdade	955	
t de Student	11,7509	
valor p	< 0,0001	
t crítico	1,9625	

Tabela E-22 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre diferentes prazos

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Curtas	384	-3833,6%	-10,0%	22,2%
Longas	384	-6179,3%	-16,1%	23,6%
Variadas	384	-4739,5%	-12,3%	25,0%

Tabela E-23 – ANOVA das comparações percentuais dos *upper bounds* do método proposto e do CPLEX entre diferentes prazos

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	0,73	2	0,3644	1,5466	0,2134	3,0036
Nos grupos	270,73	1149	0,2356			
Total	271,46	1151				

Tabela E-24 – Sumário de estatísticas descritivas para a ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre diferentes prazos

<i>Grupos</i>	<i>Observações</i>	<i>Soma</i>	<i>Média</i>	<i>Variância</i>
Curto	384	108914,7%	283,6%	1876,6%
Longo	384	183053,8%	476,7%	3449,5%
Variado	384	157430,3%	410,0%	3299,5%

Tabela E-25 – ANOVA das comparações percentuais dos *lower bounds* do método proposto e do CPLEX entre diferentes prazos

<i>Origem da variância</i>	<i>SQ</i>	<i>Graus de liberdade</i>	<i>MS</i>	<i>F</i>	<i>valor-p</i>	<i>F crit</i>
Entre grupos	738,45	2	369,2242	12,8418	< 0,0001	3,0036
Nos grupos	33035,74	1149	28,7517			
Total	33774,19	1151				