Universidade Federal da Bahia
Instituto de Matemática e Estatística

Programa de Pós-Graduação em Ciência da Computação

METAHEURÍSTICAS HÍBRIDAS BASEADAS EM PROGRAMAÇÃO POR RESTRIÇÕES PARA UM PROBLEMA DE CORTE BIDIMENSIONAL GUILHOTINADO COM DEFEITOS E RESTRIÇÕES DE PRECEDÊNCIA

Junot Freire dos Santos Neto

DISSERTAÇÃO DE MESTRADO

Salvador
27 de novembro de 2020

TERMO DE APROVAÇÃO

JUNOT FREIRE DOS SANTOS NETO

METAHEURÍSTICAS HÍBRIDAS BASEADAS EM PROGRAMAÇÃO POR RESTRIÇÕES PARA UM PROBLEMA DE CORTE BIDIMENSIONAL GUILHOTINADO COM DEFEITOS E RESTRIÇÕES DE PRECEDÊNCIA

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 27 de novembro de 2020

Prof. Dr. Rafael Augusto de Melo (Orientador)
Universidade Federal da Bahia

Prof. Dr. Tiago de Oliveira Januario
Universidade Federal da Bahia

Prof. Dr. Bruno de Athayde Prata
Universidade Federal do Ceará

RESUMO

Problemas de corte bidimensional (2BP, do inglês *Two-Dimensional bin packing problem*) são problemas clássicos de otimização combinatória pertencentes à classe NP-Difícil e tem aplicações em diversos setores como a indústria têxtil, metalúrgica e de vidro. Estes problemas consistem em alocar um conjunto de itens retangulares em placas retangulares maiores com tamanho padronizado, a fim de minimizar o desperdício de matéria-prima. Nesta dissertação é estudado um problema restrito de corte bidimensional guilhotinado associado à produção de vidro considerado no *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*, no qual existe a possibilidade de rotacionar itens em 90° e possuem ordem de precedência para serem produzidos, e as placas retangulares de matéria-prima podem possuir defeitos como rachaduras e trincos. São propostas como técnicas para este problema duas heurísticas gulosas randomizadas e um método de aprimoramento de soluções baseado em uma modelagem de programação lógica por restrições combinada com uma heurística gulosa randomizada. As técnicas foram combinadas em metaheurísticas *Multistart* e *GRASP* (do inglês, *Greedy Randomized Adaptive Search Procedure*) a fim de se obterem melhores soluções. Os experimentos realizados mostram que o uso do método de aprimoramento de soluções é vantajoso, e que algumas combinações de técnicas se mostram mais eficazes para determinados tipos de instâncias. Uma versão preliminar deste trabalho foi qualificada para a etapa final do *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*.

Palavras-chave: Problema de corte bidimensional, Metaheurísticas, Programação Lógica por Restrição

ABSTRACT

Two-Dimensional bin packing problems (2BP) are classic combinatorial optimization problems belonging to the NP-Hard class and have applications in several sectors such as the textile, metallurgical and glass industries. 2BP consists in allocating a set of rectangular items in larger rectangular plates with a standardized size in order to minimize the waste of raw material. In this dissertation, we approach a restricted version of the 2BP with guillotine cuts presented in the "ROADEF/EURO Challenge: Cutting Optimization Problem" (2018), in which there is a possibility to rotate items in 90° and soma itens have a order of precedence to be produced, and the rectangular plates may present defects in certain points. We propose two randomized greedy heuristics and a method for improving solutions based on a constraint programming model combined with a random greedy heuristic. The techniques are combined in Multistart and Greedy Randomized Adaptative Search Procedure (GRASP) metaheuristics in order to obtain better solutions. Computational experiments show that the use of the solution improvement method is advantageous, and that some combinations of the proposed techniques are more effective for certain types of instances. A preliminary version of this work was qualified for the final phase of "ROADEF/EURO Challenge: Cutting Optimization Problem" (2018).

Keywords: 2-Dimensional bin packing problem, Metaheuristics, Constraint Programming

SUMÁRIO

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS

- 2BP: problema de corte bidimensional (do inglês *Two-dimensional bin packing problem*)
- GRASP: procedimento de busca adaptativo guloso randomizado (do inglês *greedy randomized adaptive search procedure*)
- ILS: busca local iterativa (do inglês *iterated local search*)
- LCR: lista de candidatos restrita
- VNS: busca local variável (do inglês *variable neighborhood search*)

Capítulo 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

Problemas de otimização combinatória são amplamente estudados em diversas áreas do conhecimento devido às suas várias aplicações em atividades industriais e empresariais, que costumam envolver um alto custo financeiro e a melhoria nessas atividades pode trazer uma grande economia e aumento de produtividade. Estes problemas comumente consistem em selecionar elementos a fim de minimizar (ou maximizar) uma determinada função de custo respeitando um conjunto de restrições pré-definido. Muitos destes problemas, por sua vez, pertencem à classe NP-Difícil (Johnson & Garey, 1979), o que torna sua resolução complexa, desafiadora e algumas vezes inviável quando feita manualmente para um grande volume de dados. Planejamento do quadro de horários universitário (J. Freire & Melo, 2016), otimização de redes de transporte (Gayialis, Konstantakopoulos, & Tatsiopoulos, 2019) e escalonamento de trabalhos em máquinas (Gao et al., 2019) são alguns exemplos de problemas desta classe.

Diferentes abordagens podem ser escolhidas para se resolver instâncias de problemas de otimização combinatória. A escolha de quais as mais adequadas para cada situação vão depender das características e restrições do problema estudado. Dentre as técnicas mais utilizadas para resolução destes problemas estão os algoritmos exatos e as técnicas heurísticas.

Algoritmos exatos são aqueles que garantem que uma solução de valor ótimo seja encontrada se houver tempo de execução suficiente (Wolsey, 1998). No entanto, estes algoritmos podem ser inviáveis para alguns problemas ou para algumas instâncias de grande porte. Isso ocorre pois existem casos em que o espaço de busca de soluções é tão vasto que estes algoritmos não possuem tempo hábil para localizar a solução de melhor valor. Além disso, para alguns problemas a geração de soluções viáveis também pode ser uma tarefa difícil e que exija um tempo de processamento alto em comparação com o que seria aceitável para a atividade realizada. Um exemplo deste caso de tempo de processamento inviável para uma atividade é quando existe a necessidade de se gerar soluções para um problema diariamente com dados novos que devem ser utilizadas no

mesmo dia. Se o método exato demorar um dia completo para produzir uma nova solução, não haverá como por seus resultados na prática.

Técnicas heurísticas oferecem uma outra via para se trabalhar com estes problemas. Elas não dão uma garantia de encontrar uma solução de valor ótimo (Cormen, Leiserson, Rivest, & Stein, 2009), mas podem ter tempo de execução inferior. Em diversas situações, algumas heurísticas podem inclusive produzir soluções com bons valores em um tempo muito menor quando comparadas com os métodos exatos.

Estas heurísticas também podem ser organizadas utilizando diferentes procedimentos de forma a escapar de ótimos locais que venham a ocorrer, encontrando melhores soluções. Estes procedimentos são denominados metaheurísticas (BoussaïD, Lepagnot, & Siarry, 2013), e consistem em diferentes formas de se combinar técnicas que geram soluções iniciais com métodos de busca local ou aprimoramento de soluções.

Alguns exemplos de metaheurísticas são a busca local iterativa (ILS, do inglês, *iterated local search*), como visto em Lourenço, Martin, and Stützle (2003), o *Multistart* com busca local, como visto em Martí, Resende, and Ribeiro (2013), o procedimento de busca gulosa randomizada adaptativa (GRASP, do inglês *Greedy Randomized Adaptative Search Procedure*), como visto em Festa and Resende (2002), e algoritmos genéticos, como visto em Ge, Qiu, Wu, and Pu (2008).

Dentre os vários problemas existentes de natureza combinatória, a alocação de objetos em espaços finitos N-dimensionais está presente nas atividades humanas desde o início da industrialização, porém sua resolução não é uma tarefa trivial. Estas alocações de objetos são caracterizadas como problemas de corte e empacotamento (do inglês *bin-packing problems*), sendo problemas muito estudados na otimização combinatória.

A importância econômica destes problemas de corte e empacotamento é devida às suas várias aplicações, conforme o nível de dimensões em que se deseja cortar ou empacotar. A alocação em espaços unidimensionais possui como exemplo a seleção de barras de ferro para produção de colunas de concreto armado (Coffman Jr, Garey, & Johnson, 1996). Para espaços bidimensionais pode-se citar a produção de itens através do corte de placas planas de vidro ou metal (Morabito & Pureza, 2010) e para ambientes tridimensionais, o empacotamento de caixas em contâineres (Martello, Pisinger, & Vigo, 2000). Estas aplicações possuem uma alta complexidade de resolução, o que torna difícil a determinação de boas soluções de forma manual.

Listada como uma das aplicações acima, a produção de itens planos a partir do corte de placas planas a fim de se atender uma demanda pré-determinada de produtos é uma tarefa complexa, e que pode possuir variações a depender do tipo da indústria e do equipamento utilizado. Esta tarefa é uma aplicação prática do Problema de Corte Bidimensional (2BP, do inglês *Two-Dimensional Bin Packing Problem*), que consiste em dispor itens planos em grandes placas de forma que um mecanismo de corte possa produzi-los, com o objetivo de gerar o menor desperdício de matéria-prima possível (Polyakovsky & M'Hallah, 2009). O 2BP pertence à classe de problemas NP-Difícil, o que comprova a sua resolução como desafiadora e muitas vezes inviável de ser feita manualmente. Portanto, abordagens que gerem soluções de baixo nível de desperdício são necessárias devido à demanda constante de diminuição de custos na linha de produção de indústrias que trabalhem com este tipo de problema, como a metalúrgica, têxtil, madeireira e de vidros. De acordo com dados

apresentados em L. L. R. Freire (2016), contabilizando apenas a indústria de vidros planos, em 2014 foi produzido um volume de 65 milhões de toneladas em produtos. Isto mostra o quão vantajoso pode ser o estudo do 2BP, pois pequenas melhorias no processo de corte podem economizar toneladas de desperdício em matéria-prima.

Devido à sua aplicação em várias atividades industriais, o 2BP apresenta algumas variações para atender às mais diversas necessidades. Dentre estas variações estão os tipos de corte (guilhotinado e não-guilhotinado) e o número de alternâncias possíveis entre cortes verticais e horizontais que a máquina pode realizar. Também podem ser listadas a possibilidade de rotação de itens, a fim de permitir diferentes combinações de disposição de peças no espaço bidimensional, e a presença de avarias como trincos ou rachaduras nas placas de matéria-prima. Estas diferentes características que ajudam o problema a se adequar à realidade do mercado podem dificultar a resolução do mesmo e demandar o desenvolvimento de técnicas mais especializadas, tornando vasto o estudo do corte bidimensional, como visto em Lodi, Martello, and Monaci (2002).

Dentre as diversas técnicas utilizadas para resolver problemas de corte bidimensional, as mais comuns são: programação linear inteira (Furini, Malaguti, & Thomopulos, 2016), heurísticas baseadas em geração de colunas (Furini, Malaguti, Durán, Persiani, & Toth, 2012), programação dinâmica (Morabito & Pureza, 2010) e metaheurísticas como busca tabu (Alvarez-Valdés, Parreño, & Tamarit, 2007) e VNS (do inglês, *variable neighborhood search*, Dusberger and Raidl (2014)).

Este trabalho apresenta um estudo sobre um caso particular do problema restrito de corte bidimensional apresentado no *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*, que se baseia em linhas de produção de algumas indústrias de corte de vidro feitos por flutuação (do inglês, *float process*, [Takahashi et al. (2013)]). Este caso particular do problema possui um conjunto de características que o torna diferente das variações mais estudadas do 2BP enquanto o aproxima da realidade de algumas indústrias.

1.2 OBJETIVOS E PRINCIPAIS CONTRIBUIÇÕES

Nesta dissertação é estudado um caso particular do problema de corte bidimensional vivenciado na indústria de vidros, o que torna o seu estudo muito importante para resolver casos práticos desta área. Este problema restrito de corte bidimensional também é apresentado no *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*. O objetivo desta dissertação é propor técnicas capazes de gerar boas soluções para o problema estudado, identificando quais delas possuem maior eficácia para alguns conjuntos de instâncias.

São propostas as seguintes técnicas: duas heurísticas gulosas randomizadas com opção de escolha dentre dois critérios gulosos, e um método de aprimoramento de soluções que combina modelagem de programação por restrições com as heurísticas gulosas randomizadas. Ambos os algoritmos heurísticos são gulosos na seleção de itens e realização de cortes nas placas. Suas diferenças se baseiam na forma como a área das placas é percorrida durante o processo de produção de itens. O primeiro algoritmo percorre a área das placas intermediárias produzidas durante o processo de corte de forma iterativa,

armazenando apenas a informação dos itens produzidos em uma estrutura baseada em listas. Já o segundo algoritmo percorre a área destas placas intermediárias de forma recursiva, guardando a informação destas placas em uma estrutura baseada em florestas. O método de aprimoramento de soluções, por sua vez, tem como objetivo reorganizar os itens produzidos em cada placa a fim de aglutinar áreas não utilizadas para que novos itens possam ser produzidos. Os algoritmos heurísticos e o método de aprimoramento de soluções estão combinados em duas metaheurísticas: *Multistart* e *GRASP*, com o objetivo de obter diferentes soluções que possam ser geradas pela combinação destas técnicas a fim de obter melhores resultados. Finalmente, as técnicas propostas são avaliadas utilizando instâncias disponibilizadas pelo *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*.

Os resultados alcançados mostram que as soluções geradas possuem um padrão estável na taxa média de matéria-prima desperdiçada em um baixo tempo de execução. Foi observado que para a metaheurística *Multistart* os resultados possuem um padrão de melhora quando o tempo de execução é aumentado. O mesmo comportamento não é observado na metaheurística *GRASP*.

Uma versão preliminar desta dissertação foi publicada em J. Freire, Melo, Queiroz, Modesto, and Carvalho (2019), onde foram apresentados a heurística iterativa e o método de aprimoramento de soluções combinados em uma metaheurística *Multistart*. Além disso, os algoritmos apresentados nesta dissertação garantiram a classificação da equipe do autor para a fase final do *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*, obtendo as maiores notas dentre as equipes brasileiras na fase de classificação para as finais.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante da dissertação é organizado da seguinte forma. O Capítulo 2 traz a fundamentação teórica dos conceitos utilizados neste trabalho. O Capítulo 3 apresenta trabalhos relacionados sobre corte bidimensional. O Capítulo 4 formaliza a definição do problema. O Capítulo 5 descreve as heurísticas gulosas randomizadas, a técnica de aprimoramento baseada em uma modelagem de programação por restrições e as metaheurísticas desenvolvidas. O Capítulo 6 resume os experimentos computacionais para instâncias disponibilizadas pelo *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*. Comentários finais e propostas de trabalhos futuros são discutidos no Capítulo 7.

Capítulo

2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados alguns dos principais fundamentos teóricos nos quais a dissertação foi baseada.

2.1 TÉCNICAS HEURÍSTICAS

Para diversos problemas de natureza combinatória, o desenvolvimento de métodos que encontrem soluções ótimas a partir de qualquer instância não é uma tarefa factível. Devido ao crescimento exponencial do número de soluções viáveis para problemas da classe NP-Difícil, nem sempre é possível obter resultados de melhor valor em um limite de tempo aceitável. Desta forma, algoritmos eficientes que produzam soluções viáveis de alta qualidade para esta classe de problemas, mesmo sem haver garantia de otimalidade, são largamente utilizados (Müller-Merbach, 1981). Estes algoritmos são chamados de heurísticas.

Técnicas heurísticas que constroem soluções de forma incremental são muito utilizadas para problemas NP-Difíceis (Lahtinen, Myllymäki, Silander, & Tirri, 1996). A cada iteração destas técnicas um elemento é selecionado e adicionado na solução parcial até que uma solução completa seja obtida. Algoritmos gulosos são exemplos de métodos que geram soluções para diversos problemas desta forma, selecionando sempre o elemento que represente a melhor opção no momento. Esta estratégia de seleção leva a um ótimo local, que pode não ser equivalente a um ótimo global para qualquer instância recebida como entrada.

Algoritmos gulosos randomizados, por sua vez, tentam fugir deste ótimo local (Hart & Shogan, 1987). Estas heurísticas conseguem escapar dos ótimos locais produzidos por técnicas puramente gulosas pois selecionam uma dentre as melhores escolhas possíveis de forma aleatória a cada iteração da construção de uma solução. A caracterização de quais as melhores escolhas possíveis é definida para cada algoritmo através de um parâmetro que informa a quantidade de elementos que serão disponibilizados para a escolha gulosa randomizada.

2.2 METAHEURÍSTICAS

Metaheurísticas são métodos de alto nível que conseguem combinar heurísticas e técnicas de melhoria de soluções para obter resultados de alta qualidade, como visto em Resende and Ribeiro (2016). Estes métodos conseguem seguir além das soluções obtidas por heurísticas, obtendo resultados que não seriam alcançados de forma simples sem a sua utilização. *Multistart* (Martí et al., 2013) e procedimentos de busca gulosa randomizada e adaptativa (*GRASP*, do inglês *greedy randomized adaptive search procedures* (Festa & Resende, 2002)) são alguns exemplos de metaheurísticas muito estudadas na literatura.

Métodos *Multistart* são metaheurísticas simples que consistem em executar uma heurística randomizada por um determinado número de iterações, armazenando sempre a solução de melhor valor já encontrada. Devido à natureza aleatória das heurísticas utilizadas, a cada execução novas soluções são obtidas, o que não seria possível utilizando algoritmos gulosos comuns. A metaheurística *Multistart* itera enquanto um critério de parada não for atingido, a exemplo de número máximo de iterações ou tempo limite de execução.

Procedimentos *GRASP* são metaheurísticas muito efetivas para obtenção de boas soluções em problemas de natureza combinatória. Estes métodos consistem de um conjunto de execuções iteradas organizadas em duas etapas: geração de solução e processo de melhoria. A geração de solução pode ser feita através de uma heurística gulosa randomizada ou outro método que permita a formação de diferentes soluções. O processo de melhoria consiste na utilização de alguma técnica de aprimoramento que receba como entrada a solução gerada de forma heurística e tente melhorar seu valor através de mudanças em partes da solução. O método *GRASP* armazena a solução de melhor valor obtida em uma das iterações realizadas.

Dentre outros exemplos de metaheurísticas é possível citar *Simulated annealing* (Van Laarhoven & Aarts, 1987), busca tabu (Glover & Laguna, 1998) e algoritmos genéticos (Ge et al., 2008).

2.3 PROGRAMAÇÃO POR RESTRIÇÕES

A programação por restrições se trata de um paradigma da computação que combina a natureza declarativa da programação lógica com métodos de resolução de problemas baseados em restrições (Rossi, Van Beek, & Walsh, 2006). Este paradigma costuma ser muito utilizado para problemas de natureza combinatória, como planejamento de recursos e problemas de corte e empacotamento, e tem se mostrado eficaz em comparação com outras técnicas.

Modelos de programação por restrições são formados por um conjunto finito de restrições bem definidas e uma função objetivo a ser alcançada (Apt, 2003). As restrições combinam operadores aritméticos (soma, multiplicação, igualdade, dentre outros) e operadores lógicos (implicação, conjunção, disjunção) a fim de determinar qual o conjunto de soluções viáveis que serão aceitas pelo modelo. Este paradigma também dispõe de restrições globais (ou restrições simbólicas) que podem ser utilizadas na modelagem, a

exemplo de restrições que garantem que um conjunto de variáveis de decisão tenham valores distintos.

A função objetivo de um modelo possui a finalidade de selecionar a melhor solução possível dentre o conjunto de soluções viáveis. Esta função maximiza (ou minimiza) uma expressão aritmética com variáveis de decisão. Para alguns problemas, contudo, não é necessária a função objetivo. Nestes casos, a resolução da satisfatibilidade do modelo retorna uma solução desejada.

2.4 PROBLEMAS DE CORTE BIDIMENSIONAL

Em diversas atividades industriais, é preciso dispor (ou alocar) um conjunto de itens retangulares em regiões retangulares maiores de tamanho padronizado, de forma a otimizar um certo objetivo, como por exemplo minimizar um certo desperdício (Lodi et al., 2002). Esta alocação de itens possui diversas aplicações práticas, onde grande parte delas está voltada a indústrias como madeireiras e de vidro, na qual a disposição de itens representa a forma como os mesmos devem ser produzidos através de um processo de corte utilizando grandes placas de matéria-prima. Esta disposição deve minimizar o desperdício de matéria-prima não utilizada após o processo de corte.

Este tipo de atividade de disposição de itens planos está relacionada a um problema clássico de otimização combinatória, denominado Problema de Corte Bidimensional (2BP, do inglês *Two-Dimensional Bin Packing Problem*). O 2BP consiste em, dado um conjunto de placas retangulares, com largura e altura padronizadas, e um conjunto de itens retangulares, onde cada item possui valores de largura e altura específicos, é preciso dispor, sem sobreposição, todos os itens no menor conjunto de placas possível. Os itens serão produzidos a partir de um processo de corte realizado na matéria-prima fornecida, i.e., as placas. Sub-placas são geradas através do corte de placas ou de outras sub-placas. Sub-placas são marcadas como itens quando atingem as dimensões desejadas, enquanto sub-placas que não podem gerar mais itens são marcadas como desperdícios de matéria-prima. O 2BP é NP-Difícil, visto que é um caso mais geral do problema de corte unidimensional, que é conhecido como sendo NP-Difícil (Dyckhoff, 1990).

A partir das diversas aplicações vistas na indústria, o 2BP pode ser adaptado, e por isso possui uma vasta gama de variações. Dentre as mais comuns estão a rotação de itens, descrita na Seção 2.4.1, os tipos e limitações dos cortes, descritos na Seção 2.4.2, e a presença de defeitos e avarias na matéria-prima a ser utilizada, descrita na Seção 2.4.3.

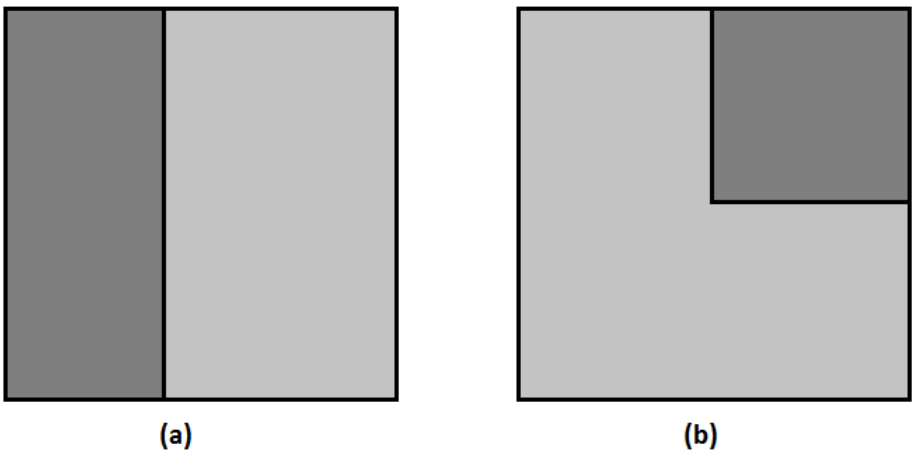
2.4.1 Rotação de itens

Em diversos trabalhos sobre corte bidimensional na literatura, os itens a ser produzidos possuem orientação fixa, ou seja, não podem ser rotacionados ao ser dispostos nas placas bidimensionais (Wuttke & Heese, 2018). Porém, também existem trabalhos onde os itens podem ser rotacionados em um ângulo de 90^{o} (Furini et al., 2012). Não se costuma permitir a rotação das placas de matéria-prima devido ao fato dessas placas possuírem dimensões muito grandes, sendo mais prática a rotação dos itens produzidos por possuírem dimensões menores.

2.4.2 Tipos de corte

Uma das principais variações do 2BP é quanto ao tipo de corte. Este pode ser classificado como guilhotinado ou não-guilhotinado. O corte guilhotinado (Dolatabadi, Lodi, & Monaci, 2012) é caracterizado como aquele que começa de um lado da placa e termina no lado paralelo oposto, e produz como resultado duas sub-placas retangulares. O não-guilhotinado (Alvarez-Valdés et al., 2007) é caracterizado como o corte no qual não é necessário cortar a placa de uma extremidade a outra, podendo resultar em sub-placas de formato não-retangular. Na Figura 2.1 é exemplificado os cortes guilhotinado (a) e não guilhotinado (b), respectivamente.

Figura 2.1: Exemplo de cortes guilhotinado e não guilhotinado



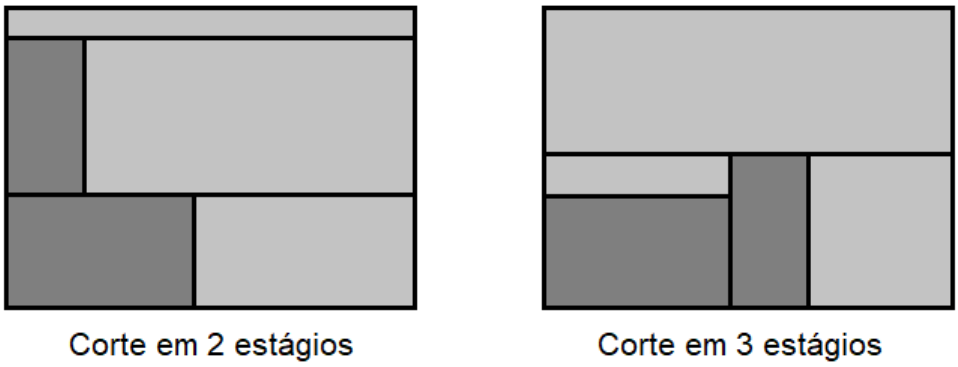
Dentre os problemas de corte guilhotinado, a produção de itens consiste em realizar cortes verticais e horizontais de forma alternada, a fim de se obter a demanda de itens nas dimensões desejadas. Em vários trabalhos da literatura esta alternância entre cortes verticais e horizontais possui uma limitação.

Um corte guilhotinado que é performado em uma placa ou sub-placa é denominado *k-cut* quando ele ele é realizado após k alternâncias entre cortes verticais e horizontais. Dentre os casos mais comuns destas limitações estão o corte até 2 estágios (Silva, Alvelos, & de Carvalho, 2010), ou 2-*cut*, e o corte em até 3 estágios (Lodi, Monaci, & Pietrobuoni, 2017), ou 3-*cut*. Outros casos de *k-cut* podem ser encontrados na literatura, como o da não limitação da alternância, presente em Furini et al. (2016). A Figura 2.2 exemplifica as diferenças do processo de corte de 2 itens utilizando limitação de 2 e 3 estágios. Neste exemplo o corte de alternância 1-*cut* é horizontal.

2.4.3 Defeitos e avarias

Além das variações apresentadas anteriormente, nem todo problema de corte possui placas em perfeitas condições. Trincos e rachaduras podem ocorrer ao final do processo de fabricação das placas, o que dificulta o processo de corte, como apresentado em Beasley

Figura 2.2: Exemplo de cortes guilhotinados em 2 e 3 estágios



(2004). Caso a ferramenta de corte atravesse um defeito (como um trinco ou rachadura) é possível que a área defeituosa se expanda ou que a placa seja quebrada. Portanto, para esta variação do 2BP, nenhum corte deve ser realizado por cima de uma região defeituosa, assim como nenhum item produzido deve conter defeitos e avarias.

Capítulo 3

TRABALHOS RELACIONADOS

Neste capítulo são apresentados trabalhos relacionados ao tema desta dissertação. Devido ao fato da variante do Problema de Corte Bidimensional (2BP) estudada ser nova e não haverem muitos trabalhos especializados nela, foram escolhidos trabalhos relacionados às diferentes versões do 2BP.

Na Seção 3.1 é feita uma revisão de trabalhos relacionados ao 2BP não-guilhotinado, e na Seção 3.2 é feita uma revisão dos trabalhos relacionados ao 2BP guilhotinado, incluindo trabalhos referentes à mesma variante estudada nesta dissertação.

3.1 PROBLEMAS DE CORTE BIDIMENSIONAL NÃO-GUILHOTINADO

Para problemas de corte não-guilhotinado, o uso de heurísticas é muito recorrente em comparação com o uso de algoritmos exatos. Em Beasley (2004) é estudado o problema de corte não-guilhotinado com presença de áreas defeituosas que não podem conter itens. Neste problema os itens a serem dispostos na placa de matéria-prima tem um valor associado e o objetivo consiste em maximizar a soma dos valores dos itens produzidos. Cada item pode ser produzido mais de uma vez, assim como podem haver itens não produzidos. Os itens não podem ser rotacionados. É apresentada uma heurística baseada em populações, onde um conjunto de soluções evolui progressivamente. A cada geração criada, as melhores soluções são armazenadas até que se chegue a um resultado desejado. Os autores utilizaram instâncias apresentadas em outros trabalhos da literatura que já possuem valores de solução ótima determinados. A otimalidade foi alcançada para todas as instâncias de pequeno porte utilizadas, enquanto que para as instâncias de grande porte foram obtidas soluções muito próximas da otimalidade.

Em Alvarez-Valdés et al. (2007) é abordada uma variação do problema abordado em Beasley (2004), onde não existe a presença de áreas defeituosas. É apresentada uma heurística gulosa para geração de padrões de corte seguida de uma busca tabu, realizando movimentos de retirada e inserção de itens nas placas que acarretem em melhorias nas soluções geradas. Para a maioria das instâncias utilizadas, as soluções apresentaram valores ótimos ou superaram outros algoritmos estado da arte comparados.

Outra abordagem de resolução do problema de corte bidimensional não-guilhotinado utilizando algoritmos bio-inspirados é apresentada em Laabadi, Naimi, El Amri, and Achchab (2019). Os autores deste trabalho propõem um algoritmo que combina algoritmos genéticos com algoritmo de busca de corvos para buscar boas soluções a partir de um conjunto inicial de soluções geradas de forma randomizada. Para validar seus resultados, os autores compararam suas soluções com as obtidas utilizando um algoritmo de otimização de enxame de partículas binárias e um algoritmo genético. Resultados mostraram que a abordagem utilizada superou o algoritmo genético em todas as instâncias testadas, mas não conseguiu superar o algoritmo de enxame de partículas para instâncias de grande porte.

Já em Wei, Hu, Lim, and Liu (2018) é proposta uma heurística baseada em *branch-and-bound* que vai construindo soluções a partir das extremidades livres da placa de matéria-prima. O método proposto é encapsulado em um *Multistart* que produz soluções enquanto houver tempo disponível e a melhor solução obtida ainda tiver possibilidade de melhoria ao ser comparada com um limite superior. Este limite superior é definido através de um modelo de programação inteira relaxado do problema. A otimalidade foi atingida para todas as instâncias pequenas e para 10% das instâncias grandes. Para instâncias muito grandes, o método proposto gera soluções de qualidade inferior em comparação com outros métodos presentes na literatura.

Também é possível encontrar trabalhos nos quais métodos exatos são utilizados para resolver o problema de corte não-guilhotinado. Em Birgin, Romão, and Ronconi (2019) é apresentado um modelo de programação linear inteira mista para uma variação do problema de corte não-guilhotinado com presença de múltiplos períodos de produção e a possibilidade de reuso de pedaços de matéria-prima não utilizados, ou seja, que não foram utilizados na produção de algum item. Neste problema os padrões de corte são realizados em múltiplos períodos para atender diferentes demandas e cada demanda de itens possui um prazo de produção a ser cumprido. As sobras de matéria-prima ao final de cada padrão de corte podem ser reaproveitadas para produções futuras através da realização de cortes guilhotinados para deixá-las em formato retangular. O modelo proposto resolve instâncias pequenas na otimalidade, porém não consegue resolver instâncias maiores.

Outra utilização de métodos exatos para resolver o problema de corte bidimensional não-guilhotinado é apresentada em Cid-Garcia and Rios-Solis (2020). Os autores deste trabalho desenvolveram um modelo de programação inteira que verifica se o conjunto de itens recebido pode ser disposto em um número de K placas de matéria-prima de mesmo tamanho. O algoritmo proposto então executa este modelo incrementando o valor de K em uma unidade sempre que o resultado da última execução do modelo retornar uma solução inviável. Foram utilizadas instâncias disponíveis na literatura, sendo que a técnica apresentada só conseguiu produzir resultados para instâncias de pequeno e médio porte. Nestes casos as soluções obtidas alcançaram a otimalidade.

3.2 PROBLEMAS DE CORTE BIDIMENSIONAL GUILHOTINADO

Diferente dos problemas citados na seção anterior, se tratando de problemas de corte guilhotinado é possível encontrar uma maior variedade de trabalhos que utilizem métodos

exatos. Em Dolatabadi et al. (2012), por exemplo, é estudada uma variante do problema de corte guilhotinado onde cada item possui um valor de lucro associado, e devem ser escolhidos itens que maximizem o total de lucro obtido que caibam na área de uma placa sem sobreposição e sem possibilidade de rotação. No trabalho são apresentados algoritmos exatos para resolução do problema: um método recursivo para o caso de existir apenas uma placa de matéria-prima a ser cortada e um modelo de programação inteira utilizando um algoritmo de *branch-and-cut* que prova a otimalidade das soluções ou identifica restrições violadas. Experimentos mostraram que a técnica *branch-and-cut* tende a ser mais rápida que o algoritmo recursivo em grande parte dos casos, embora alcance os tempos limite propostos em algumas instâncias.

Assim como no trabalho citado no parágrafo anterior, métodos exatos possuem aplicação em diversas outras variantes de problemas de corte bidimensional guilhotinado. Um exemplo do uso destes métodos é apresentado em Andrade, Birgin, and Morabito (2016). Os autores abordam uma variante com cortes em 2 estágios, sendo possível utilizar um terceiro estágio para separar um item de um desperdício. Caso uma área não utilizada da placa de matéria-prima tenha uma dimensão grande o suficiente ela pode ser utilizada em processos futuros de corte para produzir novos itens. Duas formulações de programação inteira mista foram propostas, com variações que permitem rotacionar itens e realizar o corte 1-*cut* como vertical ou horizontal.

Outros trabalhos podem ser encontrados utilizando métodos exatos para o 2BP. Em Silva et al. (2010) é apresentado um modelo de programação inteira para cortes em 2 e 3 estágios, sendo verificado que o mesmo é sensível à quantidade de itens e aos tamanhos dos itens de cada instância. Já Furini et al. (2016) apresenta um framework para modelar restrições de guilhotina como modelos de programação inteira mista, sem limitar um número máximo de estágios para realizar cortes. Macedo, Alves, and De Carvalho (2010) propôs uma modelagem de programação inteira utilizando fluxo mínimo para o corte em 2 estágios, baseando-se em casos da insdústria de madeira e conseguindo bons resultados práticos.

Métodos heurísticos também são muito utilizados para tratar com cortes guilhotinados. Lodi et al. (2017) estuda o 2BP com corte guilhotinado em até 3 estágios sem rotação de itens. É proposto um método de enumeração parcial utilizando diferentes critérios gulosos, como seleção de itens de maior área e de maior largura. Resultados indicam que este método conseguiu resolver na otimalidade 78% das instâncias consideradas.

Em Clautiaux, Sadykov, Vanderbeck, and Viaud (2017) é abordada uma variação do 2BP onde é esperada a produção de várias sequências de itens. Logo, uma área residual não utilizada ao final de um padrão de corte pode ser reutilizada em uma demanda futura. Com isso, deseja-se produzir todos os itens da demanda recebida enquanto se maximiza o comprimento desta área reutilizável ao final da última placa de matéria-prima. Os autores propuseram uma heurística de mergulho baseada em uma reformulação do 2BP como um Problema da Mochila Bidimensional. Esta heurística utiliza uma abordagem de geração de colunas e programação dinâmica para resolver o problema. Experimentos utilizando instâncias reais da indústria mostraram que esta heurística conseguiu economizar em média 1.5% da matéria-prima utilizada.

Uma variante do 2BP guilhotinado em 2 estágios é estudada em Cui and Zhao (2013),

onde é proposto um método heurístico que utiliza uma formulação de programação inteira para determinação de cortes em uma dimensão. A heurística se mostrou eficaz em resolver boa parte das instâncias na otimalidade em um curto período de tempo.

Dusberger and Raidl (2015) propõem como técnica para o 2BP guilhotinado em 3 estágios uma metaheurística VNS (do inglês *Variable Neighborhood Search*) utilizando uma proposta de "arruinar e recriar", onde partes dos padrões de corte são destruídos e depois reconstruídos utilizando heurísticas construtivas e programação dinâmica. Sua busca local consiste em destruir um número fixo de padrões de corte produzidos pelo mesmo nível *k-cut* seguida pela re-inserção dos itens removidos utilizando programação dinâmica. Esta técnica obtém melhores resultados quando os itens a serem produzidos não são muito menores que as placas de matéria-prima devido à natureza do algoritmo de reconstrução.

Em Guimarães et al. (2019) é estudado o mesmo problema abordado nesta dissertação. Foram propostas três heurísticas construtivas para o problema. A primeira consiste em uma heurística gulosa que tenta inserir os itens em orientação horizontal de acordo com o comprimento dos itens, determinando um corte 1-*cut* para cada item posicionado na parte inferior da placa de matéria-prima. A segunda consiste em uma heurística gulosa que tenta inserir vários itens lado-a-lado na parte inferior da placa utilizada para determinar um corte 1-*cut* de maior tamanho. A terceira consiste de uma heurística gulosa semelhante à primeira, porém selecionando os itens de maior área em vez dos itens de maior comprimento. De acordo com testes realizados com o conjunto de instâncias de pequeno e médio porte, fornecido pelo *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*, a terceira heurística obteve os melhores resultados, com uma taxa média de desperdício de 23.66%.

Outro trabalho que aborda o mesmo problema desta dissertação é apresentado em Libralesso and Fontan (2020). Para geração de soluções foram apresentadas duas heurísticas baseadas no algoritmo A*. Uma das heurísticas combina a técnica *branch-and-bound* com o A* para diminuir o número de nós testados ao longo de sua execução. Como esta redução pode afetar a qualidade das soluções, os autores executaram esta heurística de forma iterativa em um tipo de *Multistart*, ampliando o espaço de busca a cada iteração. A segunda técnica combina o A* com programação dinâmica, mas devido ao grande espaço de memória necessário que os autores citaram, só é possível sua utilização com instâncias pequenas. Os experimentos compararam os resultados da primeira técnica com a *Iterative Beam Search* (Golle, Rothlauf, & Boysen, 2015), mostrando que a heurística dos autores alcançou os melhores resultados em 41 das 50 instâncias, 36 instâncias a mais que o *Iterative Beam Search*. A heurística baseada em programação dinâmica se mostrou viável apenas para instâncias pequenas, e foi capaz de produzir resultados melhores que as demais.

Capítulo

4

FORMALIZAÇÃO DO PROBLEMA

Algumas linhas de produção de indústrias de corte de vidro fazem uso de uma técnica de fabricação denominada flutuação (do inglês, *float process*), onde o vidro líquido recebe um banho de estanho, é resfriado e depois moldado em grandes placas de tamanho padronizado chamadas *jumbos*. É importante salientar, no entanto, que apenas placas de tamanho padronizado que não passaram pelo processo de corte são denominadas *jumbos*. Os *jumbos* podem apresentar pequenos defeitos decorrentes do processo de flutuação, como trincos ou rachaduras. Ao final do processo de produção, um sensor analisa estas placas produzidas e mapeia possíveis defeitos existentes, que costumam ter áreas pequenas e são geralmente pontuais. Os *jumbos* são então empilhados e enviados para o processo de corte guilhotinado, a fim de gerar itens menores de acordo com as demandas dos clientes. O objetivo do problema estudado neste trabalho é gerar sequências de cortes guilhotinados (denominados padrões de corte) em *jumbos* de forma a minimizar a área total desperdiçada (isto é, pedaços de vidro cortados que não geram itens).

Seja um conjunto de *jumbos* (placas padronizadas) $\beta = \{b_1, \ldots, b_{|\beta|}\}$ de largura W e altura H, onde cada *jumbo* $b_j \in \beta$ possui um conjunto de defeitos D_{b_j}. Cada defeito $d \in D_{b_j}$ é representado por um retângulo com coordenadas de origem (x_d, y_d) (vértice inferior à esquerda), largura w_d e altura h_d. Para cada par b_j, $b_l \in \beta$, a expressão $b_j <_{cut} b_l$ significa que o *jumbo* b_j deve ser utilizado no processo de corte antes de b_l.

Seja $\mathcal{I} = \{1, \ldots, |\mathcal{I}|\}$ um conjunto de itens (denominado *batch*), onde cada item $i \in \mathcal{I}$ consiste em um retângulo de largura w_i e altura h_i. Considere $S = \{s_1, ..., s_{|S|}\}$ como um conjunto de pilhas, onde cada item $i \in \mathcal{I}$ pertence a uma única pilha $s \in S$. Para cada par de itens i e j em uma mesma pilha, a expressão $i <_{cut} j$ significa que o item i deve ser produzido no processo de corte antes do item j. Esta relação de precedência é dada apenas para itens presentes em uma mesma pilha. Não há relação de precedência de produção entre itens de pilhas diferentes. Um item i é produzido antes de um item j se os cortes que o produziram foram realizados antes dos cortes que geraram o segundo item.

Um padrão de corte consiste num conjunto de placas retangulares que compõem um *jumbo*. Uma placa *plate* possui coordenadas (x_{plate}, y_{plate}) referentes à sua posição

no *jumbo* (vértice inferior à esquerda), largura w_{plate} e altura h_{plate} e profundidade $depth_{plate} = k$ que indica que ela foi produzida com um k-*cut*. Cada *jumbo* possui profundidade $depth_{jumbo} = 0$. Placas que não sofrem mais processos de corte são marcadas como item, quando são correspondentes a um item que ainda precisa ser produzido, ou como desperdício, quando não correspondem a nenhum item que precisa ser produzido.

Considere também que a máquina que produz os padrões de corte tem limitações físicas acerca de quais os tamanhos mínimo e máximo que ela consegue cortar uma placa. Os valores $wcut_{min}$ e $hcut_{min}$ representam os tamanhos mínimos possíveis para cortes 1-*cut* e 2-*cut* consecutivos, respectivamente. Os valores $wcut_{max}$ e $desp_{min}$ representam o tamanho máximo possível para cortes 1-*cut* consecutivos e o tamanho mínimo de um desperdício, respectivamente.

Seja $P = \{p_1, p_2, \ldots, p_m\}$ uma solução viável, ou seja, um conjunto de $m \leq |\beta|$ padrões de corte que satisfazem todas as restrições do problema, onde cada padrão de corte representa a sequência de cortes e produção de itens em um determinado *jumbo*. A área total de vidro desperdiçada é calculada através da soma da área desperdiçada de cada padrão de corte em P, sendo que a placa produzida à direita do último 1-*cut* do padrão p_m é denominada área residual e não é considerada na área total desperdiçada, pois pode ser reaproveitada para futuras produções de vidro. Seja r_m o comprimento da área residual do padrão p_m, a função objetivo do problema é descrita como segue.

$$\min \; HWm - Hr_m - \sum_{i \in \mathcal{I}} w_i h_i. \tag{4.1}$$

Adicionalmente, o problema possui o seguinte conjunto de restrições:

- Todos os cortes realizados nas placas devem ser guilhotinados;
- Os cortes de nível 1-*cut* são sempre verticais, e é possível realizar cortes até o nível 3-*cut* para geração de itens finais;
- Cortes 4-*cut* são permitidos apenas para separar itens de desperdícios, não sendo possível utilizá-los para separar dois itens;
- Itens podem ser produzidos com menos de 3 cortes;
- Nenhum item produzido deve conter defeitos;
- Nenhum corte pode passar por cima de um defeito;
- Não é permitido violar a ordenação dos *jumbos*. Estes devem ser cortados na ordem que são apresentados;
- Não é permitido violar a ordenação de itens em uma pilha do *batch*;
- Os items podem ser rotacionados apenas em 90^{o}, ou seja, eles podem estar em posição vertical ou horizontal nos padrões de corte;
- Todos os items do *batch* devem ser produzidos;
- Cada item do *batch* deve ser produzido apenas uma vez, ou seja, a superprodução de itens não é permitida;

- Os *jumbos* são sempre horizontais e não podem ser rotacionados;
- A sobreposição de itens não é permitida;
- Dois cortes 1-*cut* consecutivos devem respeitar uma largura mínima $wcut_{min}$ e não ultrapassar uma largura máxima $wcut_{max} < W$, com exceção de um corte que gera uma peça residual;
- Cada padrão de corte deve ter ao mínimo um 1-*cut*;
- Dois cortes 2-*cut* consecutivos devem respeitar uma altura mínima $hcut_{min}$;
- A medida mínima de altura ou largura de um desperdício deve ser de $desp_{min}$.

Capítulo 5

TÉCNICAS PROPOSTAS

Neste capítulo são descritas as técnicas propostas para gerar soluções para o problema trabalhado. Como o Problema de Corte Bidimensional (2BP) e suas variações pertencem à classe NP-Difícil e há a necessidade de geração de soluções em um curto tempo de execução, foram desenvolvidas técnicas heurísticas e um método de aprimoramento de soluções para o problema, combinadas na forma de duas metaheurísticas. As notações utilizadas para representar as soluções são descritas na Seção 5.1. As heurísticas gulosas randomizadas são apresentadas na Seção 5.2. O método de aprimoramento com base em uma modelagem de programação por restrições é apresentado na Seção 5.3. A combinação das heurísticas com o método de aprimoramento em diferentes metaheurísticas é apresentada na Seção 5.4.

5.1 REPRESENTAÇÃO DE SOLUÇÕES

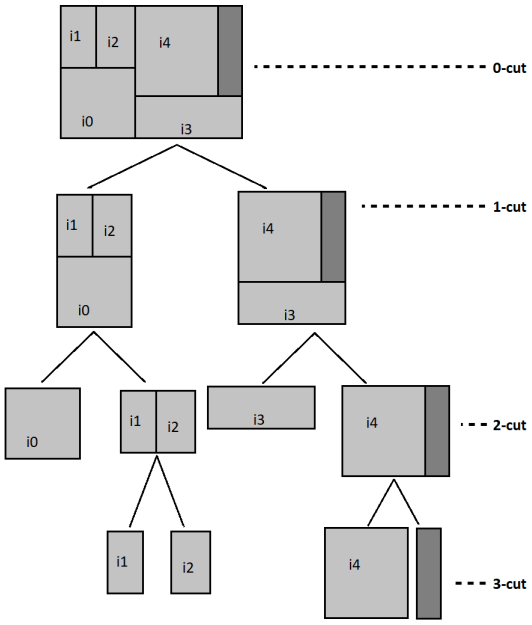
5.1.1 Notação em florestas

A estrutura principal utilizada para armazenar e representar as soluções do problema é constituída por uma floresta. Cada padrão de corte é representado por uma árvore (Dusberger & Raidl, 2014), e a solução formada por diversos padrões de corte constitui uma floresta. Para cada árvore, a raiz corresponde a placa inicial (o *jumbo*), e as folhas podem ser itens finais, desperdícios ou a área residual. Os filhos de um nó nível k-*cut* são obtidos através de cortes $(k+1)$-*cut*. A ordem de produção dos itens é obtida através de uma busca em profundidade partindo do nó raiz de cada árvore. Assume-se que a raiz da árvore possui nível 0-*cut*. A Figura 5.1 exemplifica a representação em árvore de um padrão de corte.

5.1.2 Notação de posição

Além da notação em florestas, é definida uma representação mais simples para as soluções, que consiste em uma lista de coordenadas dos itens produzidos no processo de corte (Pisinger & Sigurd, 2007). Cada tupla (i, j, x, y, r) desta lista corresponde a

Figura 5.1: Representação em árvore de um padrão de corte



um item i produzido em um *jumbo* j. Os valores x, y correspondem à posição que o item está disposto no *jumbo*, e r corresponde à orientação da peça (rotacionada ou não rotacionada). É adotada como posição do item o vértice da posição inferior esquerda do mesmo.

5.2 HEURÍSTICAS GULOSAS RANDOMIZADAS

Heurísticas gulosas randomizadas são algoritmos gulosos com um componente randomizado, ou seja, em vez de selecionar sempre a melhor escolha em um dado momento, o algoritmo escolhe uma dentre as melhores escolhas no momento. Isto permite que diferentes soluções sejam alcançadas, permitindo escapar de ótimos locais.

Duas heurísticas gulosas randomizadas foram desenvolvidas a fim de gerar soluções iniciais para o problema. Ambas utilizam os mesmos critérios gulosos e variam quanto à forma que as placas são cortadas e armazenadas em memória: de forma iterativa e recursiva. A Subseção 5.2.1 descreve os critérios gulosos utilizados pelas heurísticas. A heurística iterativa está descrita na Subseção 5.2.2, já na Subseção 5.2.3 é descrita a heurística recursiva.

5.2.1 Critérios gulosos

Dois critérios gulosos para seleção de itens são disponibilizados para as heurísticas desenvolvidas. Estes critérios foram baseados nos apresentados em Lodi et al. (2017) por terem produzido bons resultados:

- Maior dimensão: seleção do item $i \in \mathcal{I}$ que possua o maior valor calculado através da função $g_1(i) = max(w_i, h_i)$;

- Melhor encaixe de um item em uma região retangular de largura w e altura h: seleção do item $i \in \mathcal{I}$ que possua o menor valor calculado através da função $g_2(i, w, h) = w - w_i + h - h_i$.

5.2.2 Heurística iterativa

A primeira heurística proposta consiste em um método guloso randomizado que identifica qual item deve ser inserido em uma placa retangular e realiza os cortes guilhotinados necessários, gerando até 2 novas placas para corte. As placas que forem geradas pelos processos de corte são utilizadas pela heurística de forma iterativa.

A produção de itens em um *jumbo* é descrito no Algoritmo 1. O algoritmo recebe como parâmetros o *jumbo* b_j no qual os itens serão produzidos, o conjunto de itens a ser produzidos $\mathcal{I}$ e um parâmetro numérico k utilizado pelo critério guloso.

Algoritmo 1: CORTA-JUMBO-ITERATIVO(b_j, $\mathcal{I}$, k)

```
1  I_pr ← ∅;
2  S ← {b_j};
3  enquanto S ≠ ∅ e I ≠ ∅ faça
4      Armazene em p o topo da pilha S, removendo o elemento de S;
5      Construa uma LCR(I, p) de tamanho k;
6      se LCR ≠ ∅
7          Selecione aleatoriamente um item i ∈ LCR utilizando uma distribuição
            uniforme;
8          Armazene em {x_i, y_i} a melhor posição para o item i;
9          Adicione a tupla {i, {x_i, y_i}} ao conjunto I_pr;
10         Remova o elemento i do conjunto de itens I;
11         Adicione as placas {x_i, y_i + h_i, w_i, h_p − h_i, b_j} e {x_i + w_i, y_i, w_p − w_i, h_p, b_j}
            à pilha S de acordo com o nível do corte produzido;
12 retorne I_pr;
```

Inicialmente é criado um conjunto vazio de itens produzidos (linha 1) e uma pilha de placas a serem cortadas que contém inicialmente apenas o *jumbo* b_j (linha 2). Em seguida é executado um laço de repetição enquanto houverem placas a serem cortadas na pilha S e enquanto houverem itens a ser produzidos (linha 3). A cada iteração deste laço a placa no topo da pilha S é selecionada para o corte. Em seguida é criada uma lista de candidatos restrita (LCR) de tamanho máximo k com os itens de $\mathcal{I}$ que cabem na placa selecionada para corte, seguindo um dos critérios gulosos (linha 5). Caso a LCR não esteja vazia, um de seus itens é selecionado aleatoriamente e a melhor posição para seu armazenamento na placa (levando em consideração os defeitos) é calculada (linhas 6-8). A melhor posição para um item é considerada como sendo o ponto mais abaixo e à esquerda da placa. Caso a região ocupada pelo item nesta posição possua algum defeito, a melhor posição para a produção do item é considerada como o ponto acima do defeito mais acima (para cortes de profundidade par) ou como o ponto à direita do defeito mais à direita (para cortes de profundidade ímpar). Este item é então inserido na lista de itens produzidos e as placas acima e à direita produzidas pelo processo de corte guilhotinado da

placa são inseridas na pilha S para serem cortadas posteriormente em ordem decrescente do nível k-*cut* em que foram produzidas (linhas 9-12). Ao final da execução do algoritmo, é retornada a lista de itens produzidas no *jumbo* b_j, representando uma solução parcial.

A geração de uma solução completa contendo toda a demanda do *batch* na quantidade de padrões de corte necessários é descrita no Algoritmo 2. O algoritmo recebe como parâmetros o conjunto de *jumbos* β, o conjunto de itens $\mathcal{I}$ e um parâmetro numérico k utilizado pelo critério guloso. Partindo de uma solução vazia (linhas 1-2), ele entra em um laço de repetição, no qual enquanto existirem itens não produzidos (linha 3), o Algoritmo 1 é chamado para gerar um padrão de corte utilizando um *jumbo* $b_j \in \beta$ (linhas 4-5). Ao final do laço de repetição, é retornado o conjunto de padrões de corte P que representa uma solução completa. Esta heurística utiliza a notação de posição para representação de soluções.

Algoritmo 2: CONSTRUTIVO-ITERATIVO(β, $\mathcal{I}$, k)

```
1 P ← ∅;
2 j ← 0;
3 enquanto I ≠ ∅ faça
4 |   j ← j + 1;
5 |   P ← P ∪ CORTA-JUMBO-ITERATIVO(b_j, I, k);
6 retorne P;
```

5.2.3 Heurística recursiva

A segunda heurística proposta consiste em um método guloso randomizado que identifica qual item deve ser inserido em uma placa retangular e realiza chamadas recursivas para as placas formadas através do corte guilhotinado para produção de novos itens.

A produção de itens em um *jumbo* é descrita em Algoritmo 3. O algoritmo recebe como parâmetros uma placa retangular a ser cortada p, o conjunto de itens a ser produzidos $\mathcal{I}$, o nível do corte a ser realizado e um parâmetro numérico k utilizado pelo critério guloso.

Inicialmente é verificado o critério de parada da recursão, que é quando não existem mais itens a serem produzidos ou se não é possível mais realizar cortes por se chegar num nível 4-*cut* (linhas 1-2). Caso o critério de parada seja atendido é retornado um conjunto vazio de itens produzidos neste estágio da recursão. Se o critério de parada não for atendido, são criados um conjunto para armazenar os itens produzidos $\mathcal{I}_{pr}$ e uma lista de candidatos restrita (LCR) de tamanho máximo k com os itens de $\mathcal{I}$ que cabem na placa selecionada para corte, seguindo um dos critérios gulosos (linhas 3-4). Caso a LCR não esteja vazia, um de seus itens é selecionado aleatoriamente e a melhor posição para seu armazenamento na placa (levando em consideração os defeitos) é calculada (linhas 5-7). A melhor posição para um item é considerada como sendo o ponto mais abaixo e à esquerda da placa. Caso a região ocupada pelo item nesta posição possua algum defeito, a melhor posição para a produção do item é considerada como o ponto acima do defeito mais acima (para cortes de profundidade par) ou como o ponto à direita do defeito mais à direita (para cortes de profundidade ímpar). O item i é então removido do conjunto de itens a serem produzidos e são criadas as placas acima e à direita do item posicionado na

região retangular utilizada para corte (linhas 8-10). As chamadas recursivas do Algoritmo 3 são realizadas em ordens diferentes a depender da profundidade do corte a ser realizado. Caso o corte seja par, é executado primeiro para a placa à direita e depois para a placa acima. Caso contrário as chamadas recursivas são chamadas na ordem inversa. O retorno das duas chamadas são então armazenadas em $\mathcal{I}_{pr}$ (linhas 11-14). Ao final é retornada a união do conjunto $\mathcal{I}_{pr}$ com o item i selecionado (linha 15). A execução deste algoritmo passando como entrada um *jumbo* como placa a ser cortada retorna no final uma solução parcial.

Algoritmo 3: CORTA-JUMBO-RECURSIVO(p, $\mathcal{I}$, *corte*, k)

```
1  se I = ∅ ou corte = 4
2      retorne ∅;
3  I_pr ← ∅;
4  Construa uma LCR(I, p) de tamanho k;
5  se LCR ≠ ∅
6      Selecione aleatoriamente um item i ∈ LCR utilizando uma distribuição
        uniforme;
7      Armazene em {x_i, y_i} a melhor posição para o item i;
8      Remova o elemento i do conjunto de itens I;
9      Armazene em p_cima a placa {x_p, y_p, w_p, h_i − h_p, jumbo_p};
10     Armazene em p_dir a placa {x_i + w_i, y_p, w_p − w_i, h_p, jumbo_p};
11     se corte for par
12         Adicione a I_pr o retorno das chamadas recursivas deste método com os
            parâmetros (p_dir, I, corte + 1, k) e (p_cima, I, corte, k), respectivamente;
13     senão
14         Adicione a I_pr o retorno das chamadas recursivas deste método com os
            parâmetros (p_cima, I, corte + 1, k) e (p_dir, I, corte, k), respectivamente;
15 retorne I_pr ∪ {{i, {x_i, y_i}}};
```

Algoritmo 4: CONSTRUTIVO-RECURSIVO(β, $\mathcal{I}$, k)

```
1  P ← ∅;
2  j ← 0;
3  enquanto I ≠ ∅ faça
4      j ← j + 1;
5      p ← {0, 0, W, H, j};
6      P ← P ∪ CORTA-JUMBO-RECURSIVO(p, I, 1, k);
7  retorne P;
```

A geração de uma solução completa contendo toda a demanda do *batch* na quantidade de padrões de corte necessários é descrita no Algoritmo 4. O algoritmo recebe como parâmetros o conjunto de *jumbos* β, o conjunto de itens $\mathcal{I}$ e um parâmetro numérico k utilizado pelo critério guloso. Partindo de uma solução vazia (linhas 1-2), ele entra em um laço de repetição, no qual enquanto existirem itens não produzidos (linha 3),

o Algoritmo 1 é chamado para gerar um padrão de corte utilizando um *jumbo* $b_j \in \beta$ (linhas 4-6). Ao final do laço de repetição, é retornado o conjunto de padrões de corte P que representa uma solução completa. Esta heurística utiliza a notação de florestas para representação de soluções.

5.3 MÉTODO DE APRIMORAMENTO DE SOLUÇÕES

O uso de técnicas de aprimoramento de soluções em problemas da categoria NP-Difícil é muito comum. Isto se deve ao fato de que heurísticas gulosas (randomizadas ou não) nem sempre produzem soluções ótimas (ou sequer de boa qualidade), onde pode se haver a possibilidade de realizar pequenas alterações na solução a fim de se proporcionar melhorias que não foram atingidas na etapa construtiva.

Para este fim é proposto um método de aprimoramento de soluções. Este método é baseado em uma formulação de programação por restrições que tenta redistribuir os itens previamente alocados em uma região retangular de forma a aglutinar as áreas desperdiçadas para permitir a inserção de novos itens com o uso dos algoritmos gulosos previamente apresentados.

Na Subseção 5.3.1 é proposto o modelo de programação por restrições, e na Subseção 5.3.2 é descrito o algoritmo de aprimoramento de soluções que utiliza a modelagem do problema.

5.3.1 Modelo de programação por restrições

O modelo de programação por restrições proposto neste trabalho tem como objetivo reorganizar os itens previamente dispostos em um *jumbo*, sem remover nenhum deles. Esta reorganização tenta aumentar a área após o 1-*cut* mais à direita do *jumbo*, aglutinando áreas desperdiçadas que estejam espalhadas pela placa. O aumento desta área pode permitir que mais itens sejam produzidos utilizando o mesmo espaço, possibilitanto o aumento da área residual da solução, e consequentemente diminuindo o total de área desperdiçada. O modelo é descrito a seguir.

Seja $\mathcal{I}_{b_j} \subseteq \mathcal{I}$ o conjunto de itens produzidos no *jumbo* $b_j \in \beta$ e as coordenadas de cada defeito $d \in D_{b_j}$ denotadas por x_d e y_d, respectivamente, com valores previamente determinados. Considere as variáveis de decisão correspondentes às cordenadas de cada item $i \in \mathcal{I}_{b_j}$ nos eixos x e y denotadas por $x_i \in [0, W - min(w_i, h_i)]$ e $y_i \in [0, H - min(w_i, h_i)]$, respectivamente. Considere também a variável de decisão $lim \in [0, W]$, que expressa a posição no eixo x do 1-*cut* mais à direita. Definem-se também as demais variáveis de decisão:

$$rot_i = \begin{cases} 1, & \text{se o item } i \text{ está rotacionado em } 90^{\text{o}}; \\ 0, & \text{caso contrário.} \end{cases}$$

$$r_{ih} = \begin{cases} 0, & \text{se o item } i \text{ está à esquerda do item } h; \\ 1, & \text{se o item } i \text{ está abaixo do item } h; \\ 2, & \text{se o item } i \text{ está à direita do item } h; \\ 3, & \text{se o item } i \text{ está acima do item } h. \end{cases}$$

$$check = \begin{cases} 1, & \text{se a solução atual respeita as restrições de corte;} \\ 0, & \text{caso contrário.} \end{cases}$$

O problema pode ser modelado como:

$$\min lim \tag{5.1}$$

$$r_{il} = 0 \wedge rot_i = 0 \rightarrow x_i + w_i \leq x_l \\ \wedge ((desp_{min} \geq x_l - (x_i + w_i)) \vee (0 = x_l - (x_i + w_i))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.2}$$

$$r_{il} = 0 \wedge rot_i = 1 \rightarrow x_i + h_i \leq x_l \\ \wedge ((desp_{min} \geq x_l - (x_i + h_i)) \vee (0 = x_l - (x_i + h_i))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.3}$$

$$r_{il} = 1 \wedge rot_i = 0 \rightarrow y_i + h_i \leq y_l \\ \wedge ((desp_{min} \geq y_l - (y_i + h_i)) \vee (0 = y_l - (y_i + h_i))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.4}$$

$$r_{il} = 1 \wedge rot_i = 1 \rightarrow y_i + w_i \leq y_l \\ \wedge ((desp_{min} \geq y_l - (y_i + w_i)) \vee (0 = y_l - (y_i + w_i))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.5}$$

$$r_{il} = 2 \wedge rot_l = 0 \rightarrow x_l + w_l \leq x_i \\ \wedge ((desp_{min} \geq x_i - (x_l + w_l)) \vee (0 = x_i - (x_l + w_l))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.6}$$

$$r_{il} = 2 \wedge rot_l = 1 \rightarrow x_l + h_l \leq x_i \\ \wedge ((desp_{min} \geq x_i - (x_l + h_l)) \vee (0 = x_i - (x_l + h_l))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.7}$$

$$r_{il} = 3 \wedge rot_l = 0 \rightarrow y_l + h_l \leq y_i \\ \wedge ((desp_{min} \geq y_i - (y_l + h_l)) \vee (0 = y_i - (y_l + h_l))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.8}$$

$$r_{il} = 3 \wedge rot_l = 1 \rightarrow y_l + w_l \leq y_i \\ \wedge ((desp_{min} \geq y_i - (y_l + w_l)) \vee (0 = y_i - (y_l + w_l))) \ \forall i, l \in \mathcal{I}_{b_j}, \tag{5.9}$$

$$rot_i = 0 \rightarrow lim \geq x_i + w_i \ \forall i \in \mathcal{I}_{b_j}, \tag{5.10}$$

$$rot_i = 1 \rightarrow lim \geq x_i + h_i \ \forall i \in \mathcal{I}_{b_j}, \tag{5.11}$$

$$r_{il} = 0 \wedge rot_i = 0 \rightarrow x_i + w_i \leq x_l \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.12}$$

$$r_{il} = 0 \wedge rot_i = 1 \rightarrow x_i + h_i \leq x_l \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.13}$$

$$r_{il} = 1 \wedge rot_i = 0 \rightarrow y_i + h_i \leq y_l \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.14}$$

$$r_{il} = 1 \wedge rot_i = 1 \rightarrow y_i + w_i \leq y_l \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.15}$$

$$r_{il} = 2 \wedge rot_l = 0 \rightarrow x_l + w_l \leq x_i \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.16}$$

$$r_{il} = 2 \wedge rot_l = 1 \rightarrow x_l + h_l \leq x_i \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.17}$$

$$r_{il} = 3 \wedge rot_l = 0 \rightarrow y_l + h_l \leq y_i \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.18}$$

$$r_{il} = 3 \wedge rot_l = 1 \rightarrow y_l + w_l \leq y_i \ \forall i \in \mathcal{I}_{b_j}, \ l \in D', \tag{5.19}$$

$$check = 1, \tag{5.20}$$

$$\text{CHECA-VIABILIDADE}(x, y, rot, check, \mathcal{I}). \tag{5.21}$$

A função objetivo (5.1) minimiza o valor da variável *lim* no *jumbo* aprimorado. As restrições (5.2)-(5.9) garantem que não exista sobreposição entre itens e que a área desperdiçada entre dois itens respeite o valor mínimo $desp_{min}$. As restrições (5.10)-(5.11) determinam que o valor da variável *lim* esteja sempre após o final do o item alocado mais a direita de b_j. As restrições (5.12)-(5.19) garantem que não haja sobreposição entre itens

e defeitos. A restrição (5.20) assegura que a variável *check* é igual a 1, isto é, o modelo é inicializado assumindo que existe uma solução viável.

A restrição (5.21) adiciona uma função de validação das soluções geradas, descrita no Algoritmo 5. Este algoritmo recebe como entrada as variáveis x, y, rot e *check*, e o conjunto $\mathcal{I}$ de itens presentes no jumbo. Inicialmente ele cria um conjunto de tuplas $\mathcal{I}_{pr}$ contendo as coordenadas e a rotação de cada item, ordenado de acordo com os valores de x (linhas 1-4). Em seguida é verificado para cada item se existem cortes guilhotinados viáveis para produção do mesmo. Caso algum item não consiga ser produzido através de cortes guilhotinados, a variável *check* recebe o valor 0 (linhas 5-7). Por fim, é verificada a precedência de itens. Caso dois itens produzidos violem a precedência obrigatória entre si, a variável *check* recebe o valor 0 (linhas 8-11). Caso esta variável de validação tenha o valor alterado em decorrência da violação de alguma das condições verificadas pelo Algoritmo 5, a solução parcial gerada é descartada pelo modelo e uma nova solução é produzida.

Algoritmo 5: CHECA-VIABILIDADE(x, y, rot, *check*, $\mathcal{I}$)

```
1  I_pr ← ∅;
2  para cada i ∈ I faça
3      Adicione em I_pr a tupla {x_i, y_i, rot_i};
4  Ordene de forma crescente I_pr de acordo com o eixo x;
5  para cada i ∈ I_pr faça
6      se não existe corte guilhotinado viável
7          check ← 0;
8  para cada i ∈ I_pr faça
9      para cada l ∈ I_pr \ {i} faça
10         se i foi produzido depois de l e i <_cut l
11             check ← 0;
```

5.3.2 Algoritmo de aprimoramento de soluções

O uso do modelo de programação por restrições a fim de realizar melhorias em soluções do problema é descrito no Algoritmo 6. O algoritmo recebe como entrada uma solução representada pelo conjunto de padrões de corte P em notação de posição, o conjunto de itens produzidos $\mathcal{I}$, uma heurística construtiva (dentre as apresentadas neste capítulo) e uma lista de jumbos $\beta' \in P$ que devem ser passada para o modelo a fim de se gerar melhorias. A lista β' é passada por parâmetro pois podem existir *jumbos* com uma taxa de área desperdiçada muito baixa ou podem haver limitações de tempo de execução que impeçam que o modelo possa ser executado para todos os padrões de corte da solução. Caso a solução que se deseje ser aprimorada esteja na notação de florestas, uma conversão para a notação de posições deve ser realizada antes da execução do algoritmo de aprimoramento.

Algoritmo 6: APLICA-MELHORIA(P, $\mathcal{I}$, $HEURISTICA$, β')

```
1  P' ← ∅;
2  para cada jumbo ∈ P faça
3      Remova de I os items do jumbo;
4      se jumbo ∈ β'
5          Execute o modelo para o jumbo;
6          se jumbo não é o último de P
7              Armazene em res a área após o último 1-cut;
8              Execute HEURISTICA para a placa res e o conjunto de itens I;
9              Remova de I todos os itens produzidos em res;
10             Armazene em jumbo os itens produzidos em res;
11     Adicione jumbo no conjunto P';
12 Retorne P';
```

A execução do método de melhoria consiste em reconstruir o padrão de corte, reorganizando os *jumbos* através do modelo. Inicialmente é criado um novo padrão de corte vazio (linha 1) e é executado um laço de repetição para cada *jumbo* no padrão de corte P (linha 2). A cada iteração, os itens da placa visitada são removidos do conjunto $\mathcal{I}$ e é verificado se a mesma está marcada em β'. Caso o *jumbo* esteja em β', é executado o modelo para ele a fim de reorganizar seus itens (linhas 4-5). Em seguida, caso o *jumbo* que recebeu a melhoria não seja o último da solução, a heurística gulosa recebida por parâmetro é executada para a área após o último 1-*cut* da placa, consideranto todos os itens presentes em $\mathcal{I}$. Os itens que forem produzidos nesta região são então removidos de $\mathcal{I}$ (linhas 6-10). Ao final de cada iteração o *jumbo* é adicionado na solução reconstruída P' (linha 11). Esta solução reconstruída é retornada ao final da execução do algoritmo (linha 12).

5.4 METAHEURÍSTICAS

A fim de se obter melhores soluções, foram desenvolvidas duas metaheurísticas que combinem as heurísticas gulosas randomizadas com o método de aprimoramento de soluções. Estas metaheurísticas se aproveitam da natureza randomizada das heurísticas desenvolvidas para percorrer um conjunto maior de soluções a fim de se obter melhores resultados. Na Subseção 5.4.1 é apresentada a metaheurística *Multistart*, enquanto na Subseção 5.4.2 é descrita a metaheurística *GRASP*.

5.4.1 Multistart

A metaheurística *Multistart*, descrita no Algoritmo 7, tem como princípio a criação de várias soluções utilizando uma das heurísticas através de um laço de repetição. O laço executa enquanto uma condição de parada não for atingida, como tempo limite de execução ou número de iterações, e armazena a melhor solução descoberta até o momento.

Ao final do laço, a melhor solução encontrada é passada para o método de aprimoramento de soluções e seu resultado é retornado.

5.4.2 GRASP

A metaheurística *GRASP* (do inglês, *Greedy Randomized Adaptative Search Procedure*) é descrita no Algoritmo 8, e possui um comportamento um pouco distinto da metaheurística *Multistart*. O Algoritmo 8 é formado por dois laços de repetição, um externo e outro interno. O laço interno produz soluções gulosas randomizadas de forma iterativa, armazenandos sempre a melhor solução encontrada.

O laço externo executa em sequência o laço interno de chamadas de heurística gulosa randomizada e em seguida envia a melhor solução encontrada no laço interno para o método de aprimoramento. Caso a solução aprimorada em uma iteração do laço externo seja a melhor encontrada ate o momento, ela é armazenada na variável P. Ao final da execução do algoritmo, esta variável possui a solução de melhor valor que foi encontrada na metaheurística, sendo retornada pela função.

Algoritmo 7: MULTISTART(β, $\mathcal{I}$, $HEURISTICA$, *criterio*)

```
1 P ←HEURISTICA(β, I);
2 enquanto criterio faça
3 |   P' ←HEURISTICA(β, I);
4 |   se P' é melhor que P
5 |   |   P ← P';
6 P' ← APLICA-MELHORIA(P, I, HEURISTICA, otimiza);
7 se P' é melhor que P
8 |   P ← P';
9 retorne P;
```

Algoritmo 8: GRASP(β, $\mathcal{I}$, $HEURISTICA$, $criterio_1$, $criterio_2$, *otimiza*)

```
1  P ←HEURISTICA(β, I);
2  enquanto criterio_1 faça
3  |   P' ←HEURISTICA(β, I);
4  |   enquanto criterio_2 faça
5  |   |   P'' ←HEURISTICA(β, I);
6  |   |   se P'' é melhor que P'
7  |   |   |   P' ← P'';
8  |   P' ← APLICA-MELHORIA(P', I, HEURISTICA, otimiza);
9  |   se P' é melhor que P
10 |   |   P ← P';
11 retorne P;
```

Capítulo 6

EXPERIMENTOS COMPUTACIONAIS

Neste capítulo são descritos os experimentos computacionais realizados e a análise de seus resultados. Estes experimentos tem como objetivo avaliar quais dentre as técnicas propostas geram os melhores resultados. Todos os experimentos computacionais foram realizados utilizando uma máquina com processador Intel Core i7-4770S Quad-Core 3.10GHz, 16GB de memória, utilizando o sistema operacional Xubuntu x86-64 GNU/Linux. Para o modelo de programação por restrições foi utilizada a ferramenta IBM ILOG CP Optimizer 12.8.

6.1 INSTÂNCIAS

As instâncias utilizadas para validação dos algoritmos propostos são divididas em 3 grupos (A, B e X) e estão disponíveis na página do *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*. O grupo A possui 20 instâncias de pequeno e médio porte, possuindo entre 5 e 392 itens. O grupo B possui 15 instâncias de médio e grande porte, possuindo entre 68 e 656 itens. O grupo X possui 15 instâncias de grande porte, possuindo entre 124 e 412 itens.

Para todas as 50 instâncias, os valores de tamanho das placas de matéria-prima e distâncias mínima e máxima de corte são definidos a seguir: $W = 6000$, $H = 3210$, $wcut_{min} = 100$, $wcut_{max} = 3500$, $hcut_{min} = 100$ e $desp_{min} = 10$.

6.2 MÉTRICA PARA AVALIAÇÃO DE SOLUÇÕES

Para fins de comparação de diferentes soluções, o valor da função objetivo (4.1) do problema será apresentado como a porcentagem da área desperdiçada em uma solução. Ou seja, dada uma solução representada por um padrão de corte P, com m *jumbos* e conjunto de itens $\mathcal{I}$, a porcentagem de área desperdiçada é dada por:

$$(HWm - Hr_m - \sum_{i \in \mathcal{I}} w_i h_i)/(HWm - Hr_m). \tag{6.1}$$

6.3 CONFIGURAÇÕES DOS PARÂMETROS

A fim de se determinar os melhores valores de parâmetros para realização dos experimentos finais, foi realizada uma etapa preliminar de experimentos. Foram considerados os seguintes valores para cada parâmetro:

- Critérios gulosos: maior dimensão do item e melhor encaixe do item em uma região retangular;
- Critérios de randomização: $k = \{2, 3, 4\}$, $\alpha = \{0\%, 5\%, 10\%\}$;
- *Jumbos* a serem otimizados no modelo de aprimoramento: $[1; m]$, $[m/2; m]$ (sendo m o número de *jumbos* utilizados na solução).

Para verificação destes valores foi utilizada a metaheurística *Multistart* considerando o número de iterações como critério de parada. Esta metaheurística foi escolhida para verificar o comportamento produzido com cada valor de parâmetro testado ao longo de várias soluções geradas. Para os testes utilizando α como critério de randomização, os valores deste parâmetro foram convertidos para notação de k considerando quantos elementos possuem até $\alpha\%$ de diferença do melhor elemento determinado de forma gulosa.

Foi selecionado um subconjunto de 7 instâncias, sendo 3 do grupo A, 2 do grupo B e 2 do grupo X. A metaheurística *Multistart* foi executada para 1000, 10000 e 50000 iterações. Após estes experimentos foi verificado quais valores de parâmetros levaram a melhores soluções. Para um dos parâmetros houve empate em relação ao número de melhores resultados, sendo necessária uma segunda etapa de verificação.

Esta segunda etapa consistiu no uso da mesma metaheurística *Multistart* seguindo o mesmo número de iterações da etapa anterior. No entanto, foram fixados os melhores valores de parâmetros já observados e uma nova amostragem de instâncias foi realizada, consistindo de 2 do grupo A, 2 do grupo B e 1 do grupo X. A segunda etapa não apresentou novos empates.

Os melhores valores de parâmetro obtidos através dos experimentos preliminares foram: critério guloso de maior dimensão, critério de randomização $k = 2$ e a segunda metade dos *jumbos* das soluções a serem otimizadas pelo método de aprimoramento ($[m/2; m]$). Os experimentos preliminares também concluíram que a execução do método de aprimoramento de soluções apresentou tempo de execução máximo de 20 segundos por *jumbo*. Portanto, o tempo limite de execução do modelo de programação por restrições foi definido para 20 segundos.

6.4 CONFIGURAÇÕES DOS EXPERIMENTOS

A fim de se avaliar o comportamento das heurísticas e como as soluções podem ser melhoradas através do método de aprimoramento, foram consideradas as seguintes abordagens nos experimentos:

- Execução da metaheurística *Multistart* utilizando a heurística iterativa para geração de soluções, denominada *MS-I*;

- Execução da metaheurística *Multistart* utilizando a heurística recursiva para geração de soluções, denominada *MS-R*.

- Execução da metaheurística *GRASP* utilizando a heurística iterativa para geração de soluções e o método de aprimoramento para melhoria das soluções, denominada *GRASP-I*;

- Execução da metaheurística *GRASP* utilizando a heurística recursiva para geração de soluções e o método de aprimoramento para melhoria das soluções, denominada *GRASP-R*;

- Execução da metaheurística *Multistart* utilizando a heurística iterativa para geração de soluções e o método de aprimoramento para melhoria das soluções, denominada *MS-I**;

- Execução da metaheurística *Multistart* utilizando a heurística recursiva para geração de soluções e o método de aprimoramento para melhoria das soluções, denominada *MS-R**;

As abordagens *MS-I** e *MS-R** consistem em versões aprimoradas das abordagens *MS-I* e *MS-R* devido a inclusão do método de aprimoramento. As abordagens *GRASP-I* e *GRASP-R* não apresentaram uma contraparte sem método de aprimoramento pois estas versões seriam idênticas a *MS-I* e *MS-R*. Cada abordagem foi executada com tempos limite de 5 e 10 minutos. Além disso, os experimentos foram replicados utilizando 5 valores de *seed* para randomização. Todas as abordagens foram testadas para cada uma das 50 instâncias.

Para cada experimento utilizando a metaheurística *GRASP*, o tempo de execução é dividido entre a heurística gulosa randomizada e o método de aprimoramento de forma a ter o máximo de soluções otimizadas pelo método de aprimoramento, considerando o tempo de execução esperado deste algoritmo. Sendo m o número de *jumbos* utilizados em uma solução inicial gerada com uma heurística gulosa e t o tempo limite de execução da metaheurística, o número de iterações do GRASP (heurística seguida de método de aprimoramento) é dada por $t/(20m/2)$. O tempo disponível para a execução da heurística é dado pela diferença entre o tempo esperado de uma iteração da metaheurística e o tempo esperado de execução do método de aprimoramento. Esta divisão de tempo pode acarretar em diferentes comportamentos quando se varia o tempo limite de execução, pois o conjunto das soluções produzidas ao final de cada iteração da metaheurística pode não estar contido no conjunto das soluções produzidas utilizando um tempo limite de execução maior.

Para os experimentos utilizando a metaheurística *Multistart* e o método de aprimoramento de soluções, o tempo de execução é dividido de forma com que a melhor solução obtida pelo algoritmo construtivo tenha tempo para ser otimizada pelo método de aprimoramento. Logo, sendo m o número de *jumbos* utilizados em uma solução inicial gerada com uma heurística gulosa e t o tempo limite de execução da metaheurística, o tempo disponibilizado para as execuções sequenciais do algoritmo construtivo é dado por $t - 20m/2$.

Caso ao longo da execução da metaheurística, uma nova solução que utilize menos *jumbos* seja obtida, o tempo de execução para o algoritmo construtivo é recalculado utilizando a mesma equação. O comportamento do *Multistart* ao variar o tempo de execução também pode ser variado, pois o método de otimização de soluções pode acarretar em diferentes taxas de melhoria a depender da solução recebida como entrada.

6.5 RESULTADOS OBTIDOS

O objetivo dos experimentos consiste em verificar se uma das 6 abordagens propostas produz bons resultados para todos os grupos de instâncias, ou se existem abordagens mais adequadas para as instâncias de algum grupo. Para todas as abordagens testadas foi verificada a melhor taxa de desperdício por instância, isto é, o valor da melhor solução observada em um conjunto de experimentos, e a taxa média de desperdício por instância, isto é, a média dos valores das soluções obtidas em um conjunto de experimentos.

As Tabelas 6.1, 6.2 e 6.3 apresentam os resultados obtidos nos experimentos com tempo limite de 5 minutos para as instâncias dos grupos A, B e X, respectivamente. As informações presentes em cada coluna das tabelas são descritas como segue. A primeira coluna contém o nome das instâncias testadas, enquanto para as demais colunas são apresentados os resultados dos experimentos. Os resultados de cada abordagem são apresentados em duas colunas, contendo respectivamente as informações da melhor taxa de desperdício e taxa média de desperdício por instância. As primeiras duas colunas referem-se aos experimentos *MS-I*, seguido pelas duas colunas dos experimentos *MS-R*. As colunas seguintes apresentam em pares os resultados dos experimentos *GRASP-I*, *GRASP-R*, *MS-I** e *MS-R**M respectivamente.

As Tabelas 6.4, 6.5 e 6.6 apresentam os resultados obtidos nos experimentos com tempo limite de 10 minutos para as instâncias dos grupos A, B e X, respectivamente. As colunas destas tabelas seguem a mesma ordem das descritas no parágrafo anterior, sendo a primeira coluna contendo o identificador das instâncias e as demais colunas apresentando o melhor resultado e a média dos resultados para cada uma das 6 abordagens testadas. Em todas as tabelas, os valores em negrito se referem ao menor valor de solução encontrado para uma instância em um dado tempo de execução, considerando todas as abordagens.

6.5.1 Análise do uso do método de aprimoramento

Inicialmente foi verificado se o uso do método de aprimoramento apresenta vantagens em relação ao valor das soluções geradas. Para isto foram comparados os valores nas tabelas 6.1, 6.2, 6.3, 6.4, 6.5 e 6.6 em relação às colunas dos experimentos sem o uso do método (*MS-I* e *MS-R*) com as demais abordagens.

Um padrão de comportamento que distingue as abordagens *MS-I* e *MS-R* das demais é a melhoria no valor das soluções com o aumento do tempo. Isto se deve ao comportamento da metaheurística *Multistart* quando não há um método de otimização aplicado. O aumento no tempo de execução possibilita a verificação de um número maior de soluções produzidas pelos métodos gulosos, o que pode acarretar na descoberta de melhores resultados. As abordagens com o uso do método de aprimoramento não possuem

este padrão, pois o uso desta ferramenta pode trazer diferentes melhorias a depender da solução recebida como entrada.

Também é observado que a execução do *Multistart* simples, em comparação com as outras abordagens que possuem o uso do método de aprimoramento, chega nos melhores resultados em poucos casos. As abordagens *MS-I* e *MS-R* juntas produziram 6 dos melhores resultados nos testes de 5 minutos e 3 dos melhores resultados nos testes de 10 minutos. Em contrapartida, as abordagens *GRASP* produziram 28 e 10 dos melhores resultados nos testes de 5 e 10 minutos, respectivamente, enquanto as abordagens *Multistart* com método de aprimoramento produziram 11 e 32 dos melhores resultados nos mesmos testes.

Considerando as abordagens *MS-I* e *MS-R* separadamente, verifica-se que a versão que utiliza o algoritmo recursivo apresenta melhores resultados que a versão do algoritmo iterativo. Para os experimentos com tempo limite de 5 minutos, a abordagen recursiva gerou melhores resultados para 46 instâncias, e para o tempo limite de 10 minutos gerou melhores resultados para 41 instâncias.

Através de todas as análises realizadas acima, conclui-se que o uso do método de aprimoramento de soluções traz melhorias nos resultados obtidos. No entanto, para ambientes em que não há possibilidade de se utilizar este método devido à falta de ferramentas que processem modelos de programação por restrições, é possível utilizar as abordagens *MS-I* e *MS-R* para se obter soluções com alguma qualidade. Neste caso recomenda-se utilizar a versão com heurística recursiva devido aos melhores resultados produzidos.

6.5.2 Análise das abordagens com uso do método de aprimoramento

Após a verificação da eficácia do uso do método de aprimoramento, foi verificada qual abordagem que se utiliza desta ferramente produz os melhores resultados. Para isto foram comparados os valores nas tabelas 6.1, 6.2, 6.3, 6.4, 6.5 e 6.6 em relação às colunas das abordagens *GRASP-I*, *GRASP-R*, *MS-I** e *MS-R**.

Ao comparar os resultados dos experimentos com os dois valores de tempo limite de execução, foi visto uma diferença no padrão de comportamento entre aqueles que executaram em 5 minutos dos que executaram em 10 minutos. Analisando os resultados dos experimentos com tempo limite de 5 minutos, presentes nas Tabelas 6.1, 6.2 e 6.3, é possível observar que o grupo de instâncias A apresentou um comportamento distinto dos grupos B e X, que possuíram alguma similaridade em relação a quais técnicas geraram os melhores valores de solução.

Para o grupo A, a metaheurística *Multistart* apresentou os melhores resultados, onde das 20 instâncias, 9 tiveram a melhor solução obtida pela abordagem *MS-I** e 8 tiveram a melhor solução obtida pela abordagem *MS-R**. A metaheurística *GRASP* gerou 4 dos melhores resultados com a abordagem *GRASP-R* e 2 dos melhores resultados com a *GRASP-I*. Para os grupos B e X, a abordagem *GRASP-R* gerou os melhores resultados, onde das 15 instâncias de cada grupo, foram 11 para o grupo B e 9 para o grupo X.

Tabela 6.1: Taxas de desperdício no grupo A (em %) - 5 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
A1	22.51	22.51	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61
A2	14.87	15.59	13.16	13.29	15.86	16.53	14.04	14.04	15.35	15.86	**11.89**	12.46
A3	14.65	14.85	14.38	14.82	14.65	15.05	14.90	16.25	**13.34**	14.60	14.39	15.05
A4	18.09	18.58	17.41	17.98	17.77	18.54	20.24	20.31	17.28	17.69	**16.61**	17.51
A5	16.44	17.08	15.55	16.05	16.29	17.07	16.42	17.75	15.96	16.38	**14.94**	15.73
A6	15.83	17.01	15.27	15.71	14.74	15.58	18.10	18.83	**14.16**	15.38	14.62	15.58
A7	19.88	19.96	18.63	19.21	18.49	19.50	20.07	20.88	17.73	19.01	**17.39**	18.31
A8	20.13	20.62	19.05	19.43	**17.87**	19.01	19.58	19.80	18.33	18.81	18.34	19.13
A9	16.04	16.63	**15.77**	16.65	17.73	18.34	18.31	18.82	16.18	17.14	16.81	17.10
A10	17.00	17.57	17.19	17.24	16.04	16.57	16.49	17.84	**14.36**	15.78	15.44	15.83
A11	21.32	22.23	19.41	20.11	20.62	21.79	20.62	20.97	20.79	21.42	**18.72**	19.42
A12	15.93	16.40	15.54	16.23	14.42	14.91	15.20	16.14	**13.81**	14.22	14.14	14.53
A13	20.45	20.77	20.24	20.33	19.47	20.03	**18.69**	19.33	19.82	20.21	22.76	24.20
A14	20.59	20.77	19.49	19.91	19.20	20.09	**19.08**	19.51	19.13	19.76	21.48	23.06
A15	22.29	22.84	21.78	22.35	21.29	21.68	21.50	21.85	**21.05**	21.48	24.16	25.80
A16	16.47	17.78	17.68	18.42	18.25	19.41	21.22	21.70	**16.09**	18.35	16.17	17.85
A17	25.75	25.75	24.96	24.96	24.41	24.41	34.84	40.89	24.41	24.41	**22.17**	22.17
A18	20.55	20.66	19.80	20.09	19.70	20.63	20.18	20.84	19.88	20.33	**18.89**	19.62
A19	15.13	16.20	15.35	16.18	15.86	17.19	17.02	18.34	**14.97**	15.94	16.43	16.69
A20	16.57	16.57	16.25	16.25	15.32	15.38	**14.37**	15.70	15.32	15.32	15.32	15.32

Tabela 6.2: Taxas de desperdício no grupo B (em %) - 5 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
B1	12.47	12.70	12.14	12.35	11.31	12.54	13.00	13.45	**10.64**	11.55	10.90	11.82
B2	19.77	20.28	19.61	19.89	18.83	19.02	**18.08**	18.65	18.56	18.80	21.93	22.61
B3	20.39	20.74	18.90	19.31	19.79	20.60	**18.66**	19.02	20.02	20.69	21.81	22.41
B4	22.21	22.38	21.37	21.59	20.53	21.25	20.36	20.68	20.40	21.05	**19.65**	20.54
B5	34.13	34.13	33.32	34.04	34.51	34.56	**33.32**	34.44	34.13	36.01	37.12	39.60
B6	23.76	24.30	22.36	22.93	22.06	23.42	**21.02**	22.43	22.69	23.55	21.16	22.14
B7	14.83	15.16	**13.26**	13.37	15.92	16.41	13.35	13.87	15.58	16.06	13.76	14.16
B8	22.20	22.41	21.61	21.86	21.15	21.27	**20.21**	20.76	20.85	21.11	23.20	24.11
B9	21.13	21.44	18.80	18.95	20.92	21.07	**18.04**	18.52	20.69	21.05	20.94	21.72
B10	24.24	24.61	22.67	23.53	23.34	23.80	**22.46**	22.90	23.24	23.83	26.89	28.43
B11	23.39	23.54	21.05	21.16	22.28	22.82	**20.27**	20.97	22.46	22.71	23.47	24.46
B12	22.58	22.95	21.61	22.04	22.02	22.19	**21.20**	21.58	22.02	22.17	24.47	25.54
B13	24.24	24.51	22.83	23.34	22.31	23.48	**21.71**	22.45	22.70	24.81	24.08	25.19
B14	20.80	20.86	19.80	20.43	19.32	19.65	19.49	20.03	**18.63**	19.56	19.05	19.26
B15	21.57	21.81	19.27	19.73	20.30	20.83	**18.71**	19.48	20.54	22.68	23.73	23.76

Tabela 6.3: Taxas de desperdício no grupo X (em %) - 5 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
X1	22.59	22.71	20.91	21.17	21.65	22.03	**19.98**	20.33	22.05	22.32	23.31	24.92
X2	15.86	15.92	**12.71**	12.83	15.92	15.94	**12.71**	13.00	14.97	15.56	12.89	13.13
X3	**18.01**	18.95	18.10	18.28	18.08	18.82	18.18	18.37	18.02	18.67	18.08	18.60
X4	20.03	20.18	19.50	19.70	19.44	19.58	**18.97**	19.20	19.03	19.62	22.62	23.56
X5	20.62	21.07	19.41	20.37	20.53	20.89	19.41	19.91	19.96	20.35	**17.46**	18.74
X6	20.37	21.03	19.86	20.23	19.93	20.65	**19.38**	19.77	20.24	20.49	21.10	22.90
X7	21.76	22.50	21.23	21.76	21.63	21.93	**21.17**	21.32	21.63	21.82	22.57	24.71
X8	37.32	37.32	**35.66**	36.74	37.32	37.34	37.32	37.33	37.32	37.33	35.78	36.60
X9	24.89	25.44	23.82	24.57	24.00	24.24	**23.31**	23.61	23.69	23.87	26.85	28.36
X10	25.48	25.78	24.24	24.61	22.39	22.85	**21.94**	22.02	22.32	22.86	24.43	26.21
X11	24.27	24.75	23.28	23.66	21.55	22.31	**20.33**	21.18	21.96	22.31	25.06	26.12
X12	20.64	20.76	19.72	20.14	**18.53**	19.14	18.55	18.74	18.64	19.42	22.55	23.78
X13	23.05	23.41	21.93	22.36	22.59	23.23	**21.25**	21.66	22.93	23.16	24.31	25.18
X14	24.69	25.00	23.73	24.13	22.77	24.04	22.39	23.16	23.13	23.64	**22.02**	22.77
X15	23.02	23.44	22.15	22.71	**19.67**	21.15	19.77	20.13	21.61	21.72	22.82	25.61

Tabela 6.4: Taxas de desperdício no grupo A (em %) - 10 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
A1	22.51	22.51	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61	**8.61**	8.61
A2	14.73	14.93	11.85	12.95	15.91	16.51	14.04	14.55	15.38	15.61	**11.14**	11.61
A3	13.91	14.43	14.38	14.82	14.77	15.07	16.16	21.91	14.30	14.49	**14.14**	14.60
A4	17.46	17.86	17.41	17.98	17.39	18.68	19.96	22.55	17.21	17.52	**14.75**	16.86
A5	16.03	16.52	15.55	16.05	16.63	17.41	16.33	17.67	15.65	16.25	**15.14**	15.77
A6	15.82	15.91	15.27	15.71	**14.16**	15.72	18.24	19.55	14.62	14.93	14.29	15.24
A7	17.78	19.47	18.63	19.21	18.83	19.43	19.92	20.42	17.39	18.24	**17.11**	18.08
A8	19.95	20.24	19.05	19.43	19.10	20.02	19.72	20.00	17.71	18.58	**17.16**	18.30
A9	16.04	16.57	15.77	16.65	16.87	17.99	17.97	19.03	**14.96**	16.31	16.51	16.81
A10	17.00	17.37	17.19	17.24	15.78	16.68	16.61	17.62	**14.64**	15.54	15.66	16.16
A11	21.32	22.14	19.41	20.11	22.02	22.12	20.46	21.04	19.92	20.81	**19.28**	19.48
A12	15.93	16.15	15.54	16.22	15.33	17.47	16.21	19.30	13.52	14.06	**13.47**	14.11
A13	19.73	20.41	20.24	20.33	19.47	19.82	20.21	20.27	19.46	19.70	**19.06**	19.43
A14	20.35	20.54	19.49	19.91	19.95	20.22	19.14	19.78	19.21	19.48	**18.76**	19.35
A15	22.29	22.68	21.78	22.35	**20.93**	21.72	21.70	22.08	21.58	21.83	20.95	21.35
A16	16.47	17.51	17.68	18.42	18.92	19.95	20.51	30.30	17.42	18.30	**15.81**	17.77
A17	25.75	25.75	23.66	24.70	31.37	33.62	34.84	40.56	24.41	24.41	**22.17**	22.75
A18	19.73	20.31	19.80	20.09	20.66	20.99	20.13	21.16	**18.44**	19.58	19.07	19.54
A19	15.13	15.66	15.35	16.18	14.78	15.88	18.74	23.24	**14.24**	15.52	16.08	16.37
A20	16.57	16.57	16.25	16.25	29.73	37.56	39.32	39.94	**15.32**	15.32	**15.32**	15.32

Tabela 6.5: Taxas de desperdício no grupo B (em %) - 10 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
B1	12.20	12.57	12.14	12.35	11.79	12.47	12.57	13.67	**9.83**	11.25	11.12	11.83
B2	19.77	20.26	19.61	19.89	17.82	18.68	17.96	18.57	18.49	18.67	**17.74**	18.46
B3	20.39	20.57	18.90	19.31	20.17	20.48	19.22	19.46	20.01	20.19	**18.50**	18.85
B4	21.94	22.22	21.37	21.59	**19.48**	21.10	20.90	21.23	19.62	20.71	20.17	20.51
B5	34.13	34.13	**33.32**	34.04	34.13	34.28	34.13	34.55	34.13	34.13	34.13	36.19
B6	23.76	24.14	22.36	22.93	22.56	23.52	22.19	22.52	22.32	22.58	**20.84**	21.54
B7	14.83	15.02	13.26	13.37	15.94	16.32	13.54	13.94	15.24	15.81	**13.00**	13.26
B8	21.61	21.93	21.61	21.86	20.85	21.11	20.54	20.83	**20.33**	20.78	20.52	20.62
B9	20.99	21.10	18.80	18.95	20.66	20.95	18.61	18.88	20.56	20.81	**17.96**	18.53
B10	24.24	24.43	**22.67**	23.53	23.38	23.54	22.97	23.26	22.97	23.23	22.78	23.00
B11	22.84	23.31	21.05	21.16	22.05	22.93	20.40	20.68	22.40	22.69	**19.41**	20.13
B12	22.58	22.92	21.61	22.04	22.14	22.65	21.57	22.01	21.72	21.92	**21.35**	21.54
B13	24.24	24.49	22.83	23.34	22.73	23.70	**21.93**	22.27	22.38	24.50	24.48	25.39
B14	20.78	20.82	19.80	20.43	19.10	19.59	20.07	20.31	19.13	19.47	**18.56**	19.15
B15	21.57	21.66	19.27	19.73	20.50	21.54	**18.66**	19.64	20.55	23.14	19.16	21.39

Tabela 6.6: Taxas de desperdício no grupo X (em %) - 10 Minutos

	MS-I		MS-R		GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média	Melhor	Média
X1	22.40	22.50	20.91	21.17	22.15	22.32	20.03	20.70	21.65	21.96	**19.54**	20.21
X2	14.66	15.29	12.71	12.83	15.92	15.94	**12.52**	13.18	14.93	15.64	12.78	12.86
X3	18.01	18.81	18.10	18.28	19.18	19.21	18.18	18.38	18.35	18.59	**17.67**	17.95
X4	20.02	20.10	19.50	19.70	19.03	19.46	19.50	19.61	19.01	19.12	**18.53**	18.95
X5	20.62	20.82	19.41	20.37	20.53	20.75	18.06	19.31	19.41	19.95	**19.41**	18.84
X6	20.37	20.80	19.86	20.23	20.46	20.75	19.86	20.14	20.24	20.45	**19.28**	19.58
X7	21.76	22.36	21.23	21.76	**20.88**	21.37	20.90	21.14	21.22	21.48	21.16	21.36
X8	36.51	37.16	35.66	36.54	37.32	37.33	36.37	37.16	**34.69**	35.79	34.80	36.11
X9	24.56	25.19	23.82	24.57	23.89	24.00	**22.93**	23.39	23.31	23.65	23.31	23.40
X10	25.48	25.62	24.24	24.61	22.64	22.81	21.94	22.04	22.29	22.69	**21.59**	21.95
X11	24.27	24.52	23.28	23.66	22.35	22.58	20.97	21.44	21.56	22.11	**20.34**	21.07
X12	20.32	20.53	19.72	20.14	18.89	19.22	18.87	19.09	**18.26**	18.64	18.43	18.86
X13	22.82	23.21	21.93	22.36	22.80	23.38	21.80	22.07	22.75	22.94	**21.25**	21.47
X14	24.08	24.69	23.73	24.13	22.91	23.80	22.20	22.73	22.83	23.15	**21.47**	21.95
X15	23.00	23.18	22.15	22.71	20.59	21.71	19.73	20.54	20.92	21.32	**19.29**	20.18

Ainda analisando os resultados das tabelas 6.1, 6.2 e 6.3, para o grupo B a metaheurística *Multistart* gerou 1 dos melhores resultados com a heurística recursiva (*MS-R**) e 2 dos melhores resultados com a heurística iterativa (*MS-I**). Para o grupo X, a metaheurística *Multistart* gerou 1 dos melhores resultados com a heurística recursiva (*MS-R**) e 3 dos melhores resultados com a heurística iterativa (*MS-I**). A abordagem *GRASP-I* não obteve um grande desempenho independente do grupo de instâncias.

Em contrapartida, para os experimentos com tempo limite de 10 minutos, presentes nas Tabelas 6.4, 6.5 e 6.6, a abordagem *MS-R** gerou a maioria dos melhores resultados em todos os grupos de instâncias, sendo 14 dos melhores resultados do grupo A, 10 do grupo B e 10 do grupo X. As demais abordagens não apresentaram o mesmo padrão de boa qualidade de solução para todos os grupos, porém observou-se que abordagem *MS-I** obteve um resultado superior às que utilizam a metaheurística *GRASP*.

Além da contabilização de quais técnicas geraram o maior número de melhores resultados, foi analizado o quão longe dos melhores resultados obtidos pelos experimentos estão as soluções obtidas por cada técnica. As Figuras 6.1 e 6.2 apresentam gráficos *boxplot* informando qual a diferença percentual das soluções geradas em cada grupo de instâncias com as melhores obtidas nos experimentos, com 5 e 10 minutos de tempo limite, respectivamente, que utilizam o método de aprimoramento de soluções. Esta diferença percentual foi calculada utilizando o indicador de Desvio Percentual Relativo. Sendo x o valor de uma solução de uma instância com um tempo limite de execução e x' o valor da melhor solução da mesma instância com o mesmo tempo limite de execução, este desvio percentual é dado por $100(x - x')/x'$. Este cálculo de diferença percentual foi utilizado pois diferentes instâncias do mesmo grupo possuem valores de solução bem distintos. Logo, um valor de solução que seja bom para uma instância pode não ser tão bom para outra instância, e considerar todos estes valores em escalas diferentes em um gráfico pode mascarar comportamentos das metaheurísticas.

O *boxplot* é um tipo de gráfico para visualizar a distribuição e valores anômalos, denominados *outliers*, dos dados. Através dele é possível realizar uma disposição gráfica comparativa dos resultados obtidos pelos algoritmos utilizados. Neste gráfico, cada caixa retangular colorida representa o conjunto de resultados obtidos com uma técnica para um grupo de instâncias. As hastes horizontais abaixo e acima de cada caixa representam os valores mínimo e máximo, respectivamente, de cada técnica. A região compreendida pela caixa corresponde a 50% das observações, sendo a linha horizontal em seu interior correspondente ao valor da mediana. Pontos acima ou abaixo das hastes horizontais de cada caixa representam *outliers*. Todos os gráficos foram projetados para um intervalo de 0% a 100% de diferença percentual para facilitar a comparação entre técnicas. Para fins de explicação, cada caixa possui abaixo de si um identificador de qual abordagem ele se refere. As caixas com identificador

Através dos gráficos presentes na Figura 6.1, é possível observar que para os experimentos com tempo limite de 5 minutos, a abordagem *MS-I** mantém um comportamento estável para todos os grupos de instâncias. Ou seja, suas soluções apresentam um comportamento parecido em relação à diferença percentual das melhores soluções dos experimentos com este tempo limite. A abordagem *GRASP-I* possui comportamento similar, porém apresenta maiores valores nos grupos A e X. Os experimentos com a heurística re-

cursiva em ambas as metaheurísticas possuem comportamentos distintos ao variar o grupo de instâncias. A abordagem *GRASP-R* possui o melhor comportamento nos grupos de instâncias B e X, porém possui o pior comportamento dentre todas as técnicas no grupo A. Para a *MS-R**, o comportamento foi o oposto. Para o grupo A esta técnica obteve bons resultados, enquanto para os grupos B e X ela possuiu os piores comportamentos.

Figura 6.1: Comportamento das técnicas executadas por grupo - 5 minutos

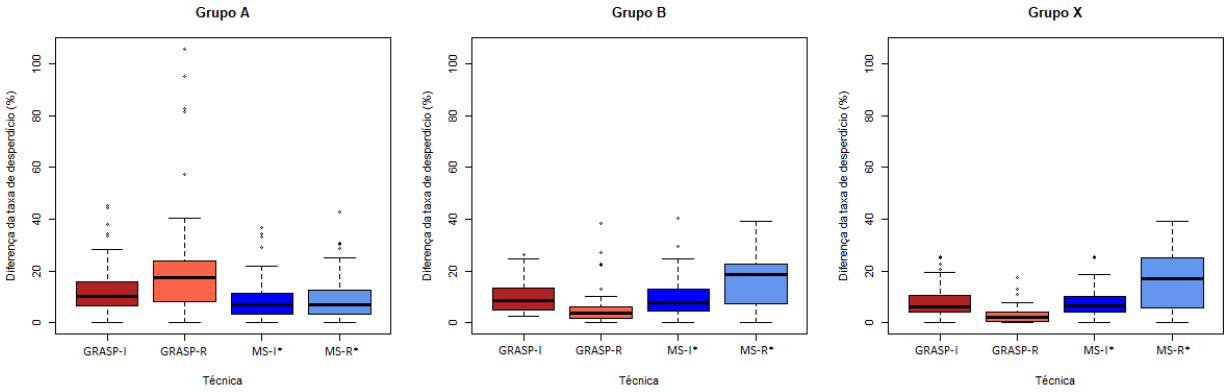
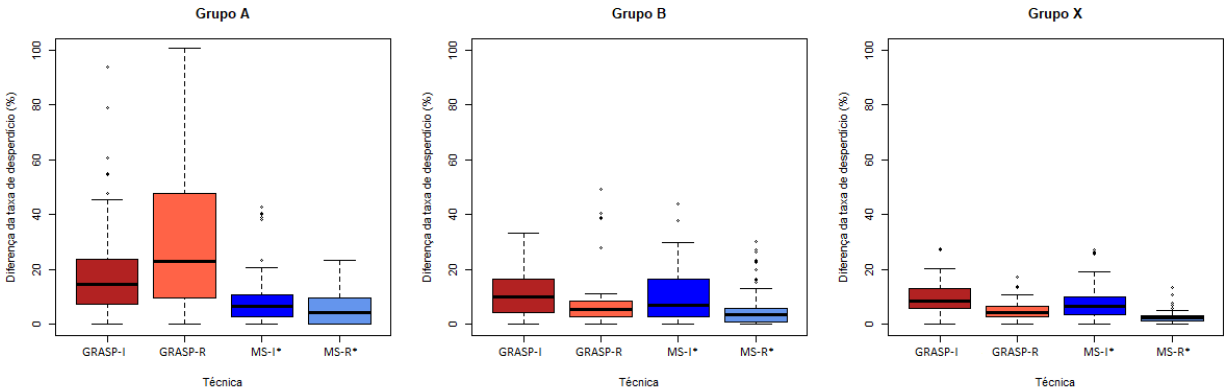


Figura 6.2: Comportamento das técnicas executadas por grupo - 10 minutos



Para os experimentos com tempo limite de 10 minutos, cujos gráficos se encontram na Figura 6.2, nota-se uma alteração no comportamento observado em algumas técnicas. A abordagem *MS-R** apresentou uma evolução no desempenho em relação aos experimentos de 5 minutos, passando a ter bons resultados em todos os grupos de instâncias e gerando melhores resultados em comparação com as outras abordagens. A abordagem *GRASP-I* por sua vez apresentou um comportamento estável para todos os grupos de instâncias, mas seus resultados ainda são piores quando comparados com as abordagens que utilizam a metaheurística *Multistart*. No entanto, a *GRASP-R* apresentou uma piora nos resultados

observados do grupo A quando comparada com os valores obtidos por ela em um tempo de execução menor. A abordagem *MS-I** produziu um bom comportamento nos grupos A e X.

Em seguida foram comparados os resultados obtidos em relação ao tempo de execução. As Tabelas 6.7 e 6.8 apresentam o desperdício médio para as melhores soluções e para todas as soluções de cada grupo, respectivamente, para cada um dos tempos de execução.

Tabela 6.7: Taxa média de desperdício dos melhores resultados por grupo (em %)

	GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	5 min	10 min	5 min	10 min	5 min	10 min	5 min	10 min
A	17.33	18.56	18.47	19.94	16.83	16.70	16.96	16.22
B	21.00	20.86	20.03	20.31	20.91	20.62	22.15	19.91
X	21.73	21.97	20.98	20.92	21.83	21.43	22.79	20.46

Tabela 6.8: Taxa média de desperdício por grupo (em %)

	GRASP-I		GRASP-R		MS-I*		MS-R*	
Instância	5 min	10 min	5 min	10 min	5 min	10 min	5 min	10 min
A	18.02	19.77	19.42	21.98	17.53	17.25	17.70	16.88
B	21.56	21.50	20.66	20.75	21.74	21.31	23.07	20.64
X	22.28	22.31	21.32	21.39	22.21	21.83	24.08	20.98

Nota-se que, em média, a metaheurística *GRASP* não apresenta um comportamento de melhoria nas soluções quando se aumenta o tempo de execução, independente da heurística construtiva utilizada. Isto ocorre pois o conjunto das soluções geradas e otimizadas utilizando esta metaheurística com o tempo de 5 minutos não está contido no conjunto das soluções geradas e otimizadas com o tempo de 10 minutos. Observa-se que o único caso onde o aumento do tempo de execução para a metaheurística *GRASP* foi benéfico na geração de soluções foi para o grupo B com a heurística iterativa. Para os demais casos desta metaheurística houve piora na solução final gerada quando foi aumentado o tempo limite de execução. Este comportamento foi observado em ambas as Tabelas 6.7 e 6.8.

Em contrapartida, a metaheurística *Multistart*, em média, apresenta melhorias em suas soluções quando o tempo de execução é aumentado. Isto acontece porque o conjunto de soluções geradas com 5 minutos está contido no conjunto de soluções geradas com 10 minutos. Logo, a solução que é passada para o algoritmo de aprimoramento de soluções considerando o tempo limite de 10 minutos é pelo menos a melhor solução gerada nos primeiros 5 minutos. No entanto, podem haver casos onde a solução final desta metaheurística com um tempo menor de execução possa ser melhor que a solução final obtida com um tempo maior, devido ao uso do método de melhoria. Isto pode acontecer pois uma solução obtida com menos iterações do *Multistart* pode haver uma melhoria final maior do que uma solução obtida com mais iterações do construtivo, mas que não possui grandes possibilidades de melhoria. Observa-se também que para os conjuntos de

instâncias B e X (com entradas de tamanho médio e grande) a melhoria nas soluções geradas quando se aumenta o tempo de execução foi maior do que para o grupo A, considerando a metaheurística *Multistart*. Este comportamento foi observado em ambas as Tabelas 6.7 e 6.8.

Capítulo 7

CONCLUSÃO

Este capítulo lista as principais contribuições desta dissertação e discute direcionamentos para possíveis trabalhos futuros. Nesta dissertação foi estudado um problema restrito de corte bidimensional guilhotinado com possibilidade de rotação de itens e presença de avarias que tem aplicação prática em inúmeras indústrias de chapas de vidro. Este problema também foi apresentado no *"ROADEF/EURO Challenge: Cutting Optimization Problem" (2018)*.

De acordo com dados apresentados em L. L. R. Freire (2016), a indústria mundial de vidros planos movimentou em 2014 um volume de 65 milhões de toneladas em produtos. Dado o grande volume de produção e a dificuldade em produzir bons padrões de corte, a busca por algoritmos que tragam melhorias no processo de corte destas placas pode resultar na economia de toneladas de matéria-prima. Com o avanço da indústria, novas particularidades também passam a ser consideradas, o que torna cria uma demanda constante para novos algoritmos que trabalhem com este tipo de problema. O problema de corte bidimensional estudado nesta dissertação apresenta uma combinação de particularidades com aplicação prática em uma vasta gama de indústrias deste ramo.

Visando trazer novas abordagens para resolução deste problema restrito de corte de placas planas, foram propostas duas heurísticas gulosas randomizadas e um método de aprimoramento de soluções que combina heurística com uma modelagem de programação lógica orientada a restrições.

As heurísticas gulosas propostas possuem muitas semelhanças em seu comportamento, como a forma de seleção de itens e a forma com que eles são cortados nas placas de vidro, respeitando um padrão na ordem de produção. Elas se diferem, no entanto, na forma como as placas são visitadas durante o processo de seleção e inserção de itens. A primeira heurística trabalha de forma iterativa, mantendo uma lista de placas não visitadas que é ordenada de acordo com o nível do corte que as produzem. Já a segunda heurística trabalha de forma recursiva, fazendo chamadas a si mesma para placa cortada.

O método de aprimoramento de soluções, por sua vez, tem como princípio a reorganização de itens que já foram atribuídos a uma placa. Este processo tenta aglutinar áreas

desperdiçadas de modo a possibilitar a produção de novos itens, diminuindo a área total desperdiçada da solução final.

Este método de aprimoramento juntamente com as heurísticas gulosas randomizadas foram encapsuladas em duas metaheurísticas, *Multistart* e *GRASP* (do inglês, *Greedy Randomized Adaptative Search Procedure*), com o objetivo de escapar de ótimos locais que possam ser atingidos através de uma única execução de cada algoritmo. Estas metaheurísticas tem como funcionamento básico a produção de várias soluções diferentes, armazenando sempre aquela que tiver o melhor valor.

A fim de se verificar qual a melhor combinação entre metaheurística, heurística gulosa randomizada e método de aprimoramento com os diferentes valores de parâmetros destas técnicas, foram realizadas baterias de testes com os 3 grupos de instâncias disponíveis para o problema. Estes testes foram realizados com dois valores de tempo de execução: 5 e 10 minutos para cada metaheurística. Através destes testes foi possível observar que o uso do método de aprimoramento acarreta em soluções que não seriam inicialmente obtidas através das heurísticas, o que levou a melhores resultados em comparação com os testes nos quais não houve uso desta técnica.

Além disso, foram observados alguns padrões de comportamento para as combinações de técnicas testadas para cada valor de tempo limite de execução. Para os testes de 5 minutos, a metaheurística *Multistart* apresentou os melhores resultados, utilizando qualquer uma das heurísticas, para o grupo de instâncias pequenas e médias. Já para os dois grupos de instâncias de médio e grande porte, a metaheurística *GRASP* com o uso da heurística recursiva apresentou os melhores resultados. Em contrapartida, para os testes de 10 minutos todos os grupos de instâncias apresentaram comportamentos semelhantes para a metaheurística *Multistart* com heurística recursiva, que foi a responsável pela maioria dos melhores resultados nesta bateria de testes.

Através destes testes e dos comportamentos observados nos grupos de soluções, espera-se que o aumento do tempo de execução gere um padrão estável no valor das soluções geradas quanto ao uso da metaheurística *Multistart*. A metaheurística *GRASP* por sua vez não apresentou bons resultados com instâncias de pequeno porte, mas para instâncias de grande porte foi possível notar uma estabilidade quanto à geração de boas soluções, comparável com o *Multistart*. Quanto às heurísticas construtivas, observou-se que a versão recursiva apresentou melhores resultados na maioria dos casos em comparação com a sua alternativa iterativa.

Diante dos resultados obtidos, é possível concluir que o uso destas técnicas pode ser uma alternativa para as indústrias que realizam este tipo de atividade mas que não possuem ferramentas de apoio a decisão para otimizar seu trabalho.

Em adição aos bons resultados encontrados pelas técnicas desenvolvidas, novos estudos podem trazer melhorias nestes algoritmos. Dentre as possíveis extensões e trabalhos futuros estão o estudo de novos métodos de aprimoramento que combinem a modelagem de programação lógica por restrições com heurísticas, como a exemplo de metodologias de "destruição e reconstrução"de soluções, o desenvolvimento de um modelo de programação inteira e a geração de limites inferiores para o problema. Outros trabalhos futuros também incluem a comparação dos métodos apresentados nesta dissertação com outras técnicas disponíveis na literatura, que abordem a mesma variante do problema

de corte bidimensional guilhotinado ou alguma variante que apresente menos restrições envolvidas.

REFERÊNCIAS BIBLIOGRÁFICAS

Alvarez-Valdés, R., Parreño, F., & Tamarit, J. M. (2007). A tabu search algorithm for a two-dimensional non-guillotine cutting problem. *European Journal of Operational Research*, *183*(3), 1167–1182.

Andrade, R., Birgin, E. G., & Morabito, R. (2016). Two-stage two-dimensional guillotine cutting stock problems with usable leftover. *International Transactions in Operational Research*, *23*(1-2), 121–145.

Apt, K. (2003). *Principles of constraint programming*. Cambridge university press.

Beasley, J. (2004). A population heuristic for constrained two-dimensional non-guillotine cutting. *European Journal of Operational Research*, *156*(3), 601–627.

Birgin, E. G., Romão, O. C., & Ronconi, D. P. (2019). The multiperiod two-dimensional non-guillotine cutting stock problem with usable leftovers. *International Transactions in Operational Research*, *27*(3), 1392-1418.

BoussaïD, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information sciences*, *237*, 82–117.

Cid-Garcia, N. M., & Rios-Solis, Y. A. (2020). Positions and covering: A two-stage methodology to obtain optimal solutions for the 2d-bin packing problem. *Plos one*, *15*(4), e0229358.

Clautiaux, F., Sadykov, R., Vanderbeck, F., & Viaud, Q. (2017). Pattern based diving heuristics for a two-dimensional guillotine cutting-stock problem with leftovers. *EURO Journal on Computational Optimization*, *7*, 265–297.

Coffman Jr, E., Garey, M., & Johnson, D. (1996). Approximation algorithms for bin packing: A survey. *Approximation algorithms for NP-hard problems*, 46–93.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. MIT press.

Cui, Y., & Zhao, Z. (2013). Heuristic for the rectangular two-dimensional single stock size cutting stock problem with two-staged patterns. *European Journal of Operational Research*, *231*(2), 288–298.

Dolatabadi, M., Lodi, A., & Monaci, M. (2012). Exact algorithms for the two-dimensional guillotine knapsack. *Computers & Operations Research*, *39*(1), 48–53.

Dusberger, F., & Raidl, G. R. (2014). A variable neighborhood search using very large neighborhood structures for the 3-staged 2-dimensional cutting stock problem. In *International workshop on hybrid metaheuristics* (pp. 85–99).

Dusberger, F., & Raidl, G. R. (2015). Solving the 3-staged 2-dimensional cutting stock problem by dynamic programming and variable neighborhood search. *Electronic Notes in Discrete Mathematics*, *47*, 133–140.

Dyckhoff, H. (1990). A typology of cutting and packing problems. *European Journal of Operational Research*, *44*(2), 145–159.

Festa, P., & Resende, M. G. (2002). GRASP: An annotated bibliography. In *Essays and surveys in metaheuristics* (pp. 325–367). Springer.

Freire, J., & Melo, R. (2016). Formulações, heurísticas e um limite combinatório para o problema de alocação de salas de aula com demandas flexíveis. *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional*, *722*, 729.

Freire, J., Melo, R., Queiroz, M., Modesto, L., & Carvalho, M. (2019). Uma heurística baseada em programação lógica por restrições para um problema restrito de corte bidimensional guilhotinado. *Anais do LI Simpósio Brasileiro de Pesquisa Operacional*, *2*.

Freire, L. L. R. (2016). A indústria de vidros planos. *Caderno Setorial ETENE*, *3*.

Furini, F., Malaguti, E., Durán, R. M., Persiani, A., & Toth, P. (2012). A column generation heuristic for the two-dimensional two-staged guillotine cutting stock problem with multiple stock size. *European Journal of Operational Research*, *218*(1), 251–260.

Furini, F., Malaguti, E., & Thomopulos, D. (2016). Modeling two-dimensional guillotine cutting problems via integer programming. *INFORMS Journal on Computing*, *28*(4), 736–751.

Gao, K., Cao, Z., Zhang, L., Chen, Z., Han, Y., & Pan, Q. (2019). A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems. *IEEE/CAA Journal of Automatica Sinica*, *6*(4), 904–916.

Gayialis, S. P., Konstantakopoulos, G. D., & Tatsiopoulos, I. P. (2019). Vehicle routing problem for urban freight transportation: A review of the recent literature. *Operational Research in the Digital Era–ICT Challenges*, 89–104.

Ge, J.-k., Qiu, Y.-H., Wu, C.-M., & Pu, G.-L. (2008). Summary of genetic algorithms research. *Application research of computers*, *10*(10), 2911–2916.

Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093–2229). Springer.

Golle, U., Rothlauf, F., & Boysen, N. (2015). Iterative beam search for car sequencing. *Annals of Operations Research*, *226*(1), 239–254.

Guimarães, M., Bogue, E., Pereira, A., Carvalho, I., Noronha, T., & Urrutia, S. (2019). Heurísticas construtivas para o problema de corte guilhotinado bidimensional em 3 estágios com restrições de precedência. *Anais do LI Simpósio Brasileiro de Pesquisa Operacional*, *2*.

Hart, J. P., & Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters*, *6*(3), 107–114.

Johnson, D., & Garey, M. (1979). *A guide to the theory of np-completeness. computers and intractability.* WH Freeman and Company.

Laabadi, S., Naimi, M., El Amri, H., & Achchab, B. (2019). A crow search-based genetic algorithm for solving two-dimensional bin packing problem. In *Joint german/austrian conference on artificial intelligence (künstliche intelligenz)* (pp. 203–215).

Lahtinen, J., Myllymäki, P., Silander, T., & Tirri, H. (1996). Empirical comparison of stochastic algorithms. Citeseer.

Libralesso, L., & Fontan, F. (2020). An anytime tree search algorithm for the 2018 ROADEF/EURO challenge glass cutting problem. *arXiv preprint arXiv:2004.00963*.

Lodi, A., Martello, S., & Monaci, M. (2002). Two-dimensional packing problems: A survey. *European journal of operational research*, *141*(2), 241–252.

Lodi, A., Monaci, M., & Pietrobuoni, E. (2017). Partial enumeration algorithms for two-dimensional bin packing problem with guillotine constraints. *Discrete Applied Mathematics*, *217*, 40–47.

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

Macedo, R., Alves, C., & De Carvalho, J. V. (2010). Arc-flow model for the two-dimensional guillotine cutting stock problem. *Computers & Operations Research*, *37*(6), 991–1001.

Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. *Operations research*, *48*(2), 256–267.

Martí, R., Resende, M. G., & Ribeiro, C. C. (2013). Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, *226*(1), 1–8.

Morabito, R., & Pureza, V. (2010). A heuristic approach based on dynamic programming and and/or-graph search for the constrained two-dimensional guillotine cutting problem. *Annals of Operations Research*, *179*(1), 297–315.

Müller-Merbach, H. (1981). Heuristics and their design: a survey. *European Journal of Operational Research*, *8*(1), 1–23.

Pisinger, D., & Sigurd, M. (2007). Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, *19*(1), 36–51.

Polyakovsky, S., & M'Hallah, R. (2009). An agent-based approach to the two-dimensional guillotine bin packing problem. *European Journal of Operational Research*, *192*(3), 767–781.

Resende, M. G., & Ribeiro, C. C. (2016). *Optimization by GRASP*. Springer.

ROADEF/EURO Challenge: Cutting optimization problem [Computer software manual]. (2018). (Online reference, http://www.roadef.org/challenge/2018/en/index.php)

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Silva, E., Alvelos, F., & de Carvalho, J. V. (2010). An integer programming model for two-and three-stage two-dimensional cutting stock problems. *European Journal of Operational Research*, *205*(3), 699–708.

Takahashi, S., Horiuchi, K., Tatsukoshi, K., Ono, M., Imajo, N., & Mobely, T. (2013). Development of through glass via (TGV) formation technology using electrical discharging for 2.5/3D integrated packaging. In *2013 IEEE 63rd electronic components and technology conference* (pp. 348–352).

Van Laarhoven, P. J., & Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications* (pp. 7–15). Springer.

Wei, L., Hu, Q., Lim, A., & Liu, Q. (2018). A best-fit branch-and-bound heuristic for the unconstrained two-dimensional non-guillotine cutting problem. *European Journal of Operational Research*, *270*(2), 448 - 474.

Wolsey, L. A. (1998). *Integer programming* (Vol. 52). John Wiley & Sons.

Wuttke, D. A., & Heese, H. S. (2018). Two-dimensional cutting stock problem with sequence dependent setup times. *European Journal of Operational Research*, *265*(1), 303–315.