UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Estudo de problemas de corte de itens irregulares com incertezas

Layane Rodrigues de Souza Queiroz

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Layane Rodrigues de Souza Queiroz

Estudo de problemas de corte de itens irregulares com incertezas

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutora em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Marina Andretta

USP – São Carlos
Março de 2022

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

S719e

Souza Queiroz, Layane Rodrigues de
Estudo de problemas de corte de itens irregulares com incertezas / Layane Rodrigues de Souza Queiroz; orientadora Marina Andretta. -- São Carlos, 2022.
142 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2022.

1. Problemas de corte e empacotamento. 2. Problemas de nesting. 3. Item irregular. 4. Otimização Estocástica. 5. Heurística. I. Andretta, Marina, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Layane Rodrigues de Souza Queiroz

Study of cutting problems of irregular shaped items with uncertainties

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Marina Andretta

USP – São Carlos
March 2022

Este trabalho é dedicado a Deus e à minha família,
fontes permanentes de amor para mim!

AGRADECIMENTOS

Agradeço a Deus, causa primária de todas as coisas, pelo dom da vida.

Agradeço meu esposo Thiago, companheiro de todas as horas, por todo o carinho, incentivo e amor.

Agradeço minha orientadora Marina pela paciência, dedicação e conselhos.

Agradeço meus familiares, amigos, professores e colegas que, de alguma forma, participaram desta caminhada comigo.

Agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pela bolsa de estudos.

"O egoísmo é a fonte de todos os vícios,
como a Caridade é a fonte de todas as virtudes."
(Allan Kardec)

RESUMO

SOUZA QUEIROZ, L. R. **Estudo de problemas de corte de itens irregulares com incertezas**. 2022. 142 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Os problemas de corte e empacotamento aparecem nas mais variadas empresas do setor logístico e de manufatura, bem como nas indústrias de móveis, vestuário, metal-mecânica, têxtil e outras. Esta tese é voltada para o estudo de problemas de *nesting*, ou seja, problemas de corte e empacotamento de itens irregulares, na presença de incertezas que surgem de contextos reais. Os problemas consideram duas dimensões e os itens são representados por polígonos convexos e/ou não convexos, enquanto os recipientes são retangulares. A primeira contribuição da tese está relacionada a duas heurísticas competitivas para o problema da mochila sem incertezas. Uma heurística é baseada no algoritmo genético de chaves aleatórias viciadas, enquanto a outra considera uma busca em vizinhança variável. Enquanto as heurísticas geram sequências de itens, três regras são usadas para o posicionamento de itens. Desenvolve-se ainda uma codificação para a solução do problema que permite ignorar posições viáveis durante o posicionamento de itens e, assim, escapar de possíveis ótimos locais. Em geral, estas heurísticas permitiram melhorar o estado-da-arte do problema, obtendo soluções cuja área ocupada aumentou em torno de 6% na média. A segunda contribuição envolve o problema de corte em faixa cuja demanda dos itens é um dado incerto. Além de propor para este problema um modelo de programação estocástica de dois estágios com recurso, apresenta-se um algoritmo *branch-and-cut* que integra uma heurística de busca em vizinhança variável para gerar soluções válidas nos nós da árvore de busca. O algoritmo proposto é competitivo com outros da literatura sobre o problema sem incertezas. No problema com incertezas, o algoritmo pode obter soluções para instâncias com até 80 cenários. Além disso, as análises das soluções do modelo de programação estocástica indicam que ignorar a aleatoriedade dos dados na escolha de uma decisão pode resultar em soluções de custo elevado. Por fim, a terceira contribuição consiste em um modelo de programação estocástica de dois estágios com recurso para um problema da mochila que apresenta defeitos no recipiente, sendo os defeitos tratados como dados incertos. As soluções geradas pelo modelo são analisadas quanto ao valor esperado da informação perfeita e o valor da solução estocástica, indicando o impacto que as incertezas têm sobre o problema. Este modelo também é extendido para considerar uma medida de risco, objetivando controlar a variabilidade das decisões de segundo estágio e, assim, obter soluções aversas ao risco. Os resultados computacionais sugerem que soluções totalmente aversas ao risco podem requerer reduções de até 28% no lucro total esperado.

Palavras-chave: problemas de corte e empacotamento, itens irregulares, incertezas, modelo de programação estocástica, heurísticas.

ABSTRACT

SOUZA QUEIROZ, L. R. **Study of cutting problems of irregular shaped items with uncertainties**. 2022. 142 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2022.

Cutting and packing problems appear in a wide variety of companies in the logistics and manufacturing sector, as well as in the furniture, clothing, metal-mechanic, textile and other industries. This thesis is focused on the study of nesting problems, that is, cutting and packing problems of irregular shaped items in the presence of uncertainties that emerge from real situations. The problems consider two dimensions and items are represented by convex and/or non-convex polygons, while the containers are rectangles. The first contribution of the thesis is related to two competitive heuristics for the knapsack problem without uncertainties. One heuristic is based on the biased random key genetic algorithm, while the other is a variable neighborhood search. While the heuristics generate sequences of items, three rules are used for item placement. An encoding is also designed for the solution of the problem that allows feasible positions to be ignored during item positioning and thus escape from possible local optima solutions. In general, these heuristics allowed to improve the state-of-the-art methods of the problem, obtaining solutions whose occupied area increased around 6% on average. The second contribution involves the strip packing problem for which the item demand is an uncertain data. In addition to proposing for this problem a two-stage stochastic programming model with recourse, a branch-and-cut algorithm is presented that integrates a variable neighborhood search heuristic to generate valid solutions at the nodes of the search tree. The proposed algorithm is competitive with others in the literature of the problem without uncertainties. In the problem with uncertainties, the algorithm can obtain solutions for instances with up to 80 scenarios. Furthermore, analyses of the solutions obtained with the stochastic programming model indicate that ignoring uncertainties in choosing a decision can result in high cost solutions. Finally, the third contribution consists of a two-stage stochastic programming model with recourse for a knapsack problem with defects in the container. The defects are treated as uncertain data. The solutions generated by the model are examined for the expected value of the perfect information and the value of stochastic solution, indicating the impact that uncertainties have on the problem. This model is also extended to consider a risk measure, aiming to control the variability of the second-stage decisions and thus obtain risk-averse solutions. Computational results suggest that fully risk-averse solutions may require reductions of up to 28% in the total expected net profit.

Keywords: cutting and packing problems, irregular shaped items, uncertainties, stochastic programming model, heuristics.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2IKP	*Two-dimensional Irregular Knapsack Problem - 2IKP*
2IKP	*Two-dimensional Irregular Knapsack Problem*
2IKP-DU	*Two-dimensional Irregular Knapsack Problem with Defects Uncertainty*
2ISP	*Two-Dimensional Irregular Strip Packing Problem*
2ISP-DU	*Two-Dimensional Irregular Strip Packing Problem with Demand Uncertainty*
ESICUP	*EURO Special Interest Group on Cutting and Packing*
EV	*Expected Value Problem*
EVPI	*Expected Value of Perfect Information*
EVV	Expectation of the Expected Value Problem
GRASP	*Greedy Randomized Adaptive Search Procedure*
GVNS	*General Variable Neighborhood Search*
irace	*Iterated Race for Automatic Algorithm Configuration*
RP	*Recourse Problem*
TOPOS	Técnicas de Optimização para o Posicionamento de Figuras Irregulares
VND	*Variable Neighborhood Descent*
VSS	*Value of Stochastic Solution*
WS	Wait-and-See

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Problemas de corte estão relacionados com o corte de objetos grandes (recipientes) para a obtenção de objetos menores (itens), buscando otimizar algum objetivo. Em indústrias de transformação como a de móveis, os recipientes estocados (por exemplo, placas de madeira) precisam ser cortados para obter itens (por exemplo, peças e componentes dos móveis). Por outro lado, problemas de empacotamento buscam pelo posicionamento de itens dentro de recipientes. Em empresas do setor logístico como a de transporte de produtos, os itens estocados (por exemplo, caixas) precisam ser empacotados em recipientes (por exemplo, contêineres que depois podem ser acoplados em caminhões ou transportados por navios).

Apesar da diferença prática entre cortar e empacotar, problemas de corte e de empacotamento apresentam uma estrutura interna similar para fins de estudo matemático e computacional. Isto possibilita tratar teoricamente um problema de corte como sendo de empacotamento e vice-versa, pois cortar um recipiente para obter itens equivale a empacotar os itens dentro do recipiente. Na literatura é comum chamá-los de problemas de corte e empacotamento, com os termos "corte" e "empacotamento" sendo considerados sinônimos (DYCKHOFF, 1990; WÄSCHER; HAUSSNER; SCHUMANN, 2007).

O estudo de problemas de corte e empacotamento é de grande relevância, pois são comumente encontrados em aplicações de empresas e indústrias, como na metal-mecânica, automotiva e de máquinas agrícolas, têxtil, de produção de móveis, transporte logístico e fretamento. Esses problemas vêm sendo estudados nas áreas de otimização e pesquisa operacional, sendo tratados como problemas de otimização combinatória, que possuem um domínio finito, porém muito grande. O interesse é obter soluções que minimizam (ou maximizam) uma função objetivo sujeita às restrições do problema tratado (SCHEITHAUER, 2017).

Em problemas de otimização combinatória tem sido impraticável a busca da melhor solução por métodos de enumeração simples ou que tentam todas as soluções possíveis, pois o número de soluções é muito grande e cresce com tamanho da entrada do problema (por

exemplo, o número de itens). Nesse ponto, existem os métodos exatos, que exploram o espaço de soluções do problema em busca da melhor solução possível (isto é, uma solução ótima) (CONFORTI; CORNUÉJOLS; ZAMBELLI, 2014). Há também as heurísticas, que exploram, de forma inteligente, parte desse espaço em busca de uma boa solução, sem garantias de encontrar a melhor possível (TALBI, 2009).

Também é de conhecimento que diversos problemas de corte e empacotamento são NP-Difíceis (GAREY; JOHNSON, 1979; DYCKHOFF, 1990), ou seja, sob a hipótese de que P é diferente de NP, não há métodos exatos de tempo polinomial no tamanho da entrada para resolvê-los, justificando, por exemplo, o desenvolvimento de heurísticas. Além disso, no contexto de itens e recipientes irregulares, Bennell e Oliveira (2009) enfatizaram que a geometria tem sido um dificultador para a proposta de métodos de solução eficazes. Outro fator dificultador pode estar relacionado aos parâmetros do problema, que nem sempre são conhecidos com exatidão e/ou *a priori*. Por exemplo, os dados podem variar em conformidade com a forma como foram recolhidos (medidos) ou não estarem completamente disponíveis no momento em que se precisa tomar as decisões, necessitando considerar a presença de possíveis incertezas no problema (BEN-TAL; NEMIROVSKI, 1998).

Diante da diversidade de problemas de corte e empacotamento, uma tipologia foi proposta por Dyckhoff (1990), sendo mais tarde estendida e revisada por Wäscher, Haussner e Schumann (2007) para também englobar problemas mais recentes. Wäscher, Haussner e Schumann (2007) consideraram cinco critérios para classificar os problemas:

- dimensionalidade: relacionada à dimensão para representar os itens e recipientes (unidimensional, bidimensional, tridimensional, ou n-dimensional);
- tipo de objetivo: maximizar a quantidade de itens produzidos (maximização da saída); minimizar o espaço não utilizado (desperdício) dos recipientes (minimização da entrada); ou mais de um objetivo;
- variedade dos recipientes: um recipiente, com nenhuma, uma, duas ou todas as dimensões variáveis; vários recipientes (idênticos ou heterogêneos) com dimensões fixas; ou recipientes não retangulares;
- variedade dos itens: itens idênticos, com poucas variações de tamanho ou bastante variados, além de poderem ter demandas similares ou bem variadas;
- forma dos itens: para os itens com duas ou mais dimensões, tem-se itens regulares (por exemplo, retângulos, círculos, esferas e paralelepípedos retângulos) ou irregulares (isto é, com forma não regular).

A partir dos critérios estabelecidos, em particular, tipo de objetivo e variedade dos itens, Wäscher, Haussner e Schumann (2007) classificaram os seguintes problemas como básicos:

- empacotamento de itens idênticos (*identical item packing problem*): dado um conjunto de itens idênticos e um número limitado de recipientes de dimensões fixas, busca-se maximizar a quantidade de itens empacotados;

- alocação (*placement problem*): dado um conjunto de itens fracamente heterogêneo e um número limitado de recipientes de dimensões fixas, busca-se maximizar o valor associado aos itens que são empacotados (ou de forma alternativa, minimizar o desperdício dos recipientes);

- mochila (*knapsack problem*): dado um conjunto de itens fortemente heterogêneo e um número limitado de recipientes de dimensões fixas, busca-se maximizar o valor associado aos itens que são empacotados;

- dimensão aberta (*open dimension problem*): dado um conjunto de itens de variedade arbitrária e um recipiente com ao menos uma dimensão variável, busca-se minimizar a parte usada da dimensão variável do recipiente;

- empacotamento em recipientes (*bin packing problem*): dado um conjunto de itens fortemente heterogêneo e recipientes de variedades arbitrárias e dimensões fixas, busca-se minimizar a quantidade de recipientes usada para empacotar todos os itens;

- corte de estoque (*cutting stock problem*): dado um conjunto de itens fracamente heterogêneo e recipientes de variedades arbitrárias e dimensões fixas, busca-se minimizar a quantidade de recipientes usada para empacotar todos os itens.

Independente do tipo de problema de corte e empacotamento, existem duas restrições que devem ser respeitadas ao procurar por uma solução viável, quais sejam: (i) quaisquer dois itens devem ser empacotados de forma a não se sobreporem (isto é, não podem ter qualquer interseção de seus interiores) e (ii) os itens devem ser empacotados de forma a ficarem inteiramente contidos nos recipientes (isto é, um item não pode ter qualquer parte sua de fora do recipiente). Existem outras restrições que surgem em contextos reais e podem ser incorporadas, como as apresentadas por Bischoff e Ratcliff (1995) e também discutidas em Bortfeldt e Wäscher (2013) e Nascimento, Queiroz e Junqueira (2021). Por exemplo, permitir a rotação dos itens, satisfazer o balanceamento de peso dentro do recipiente, agrupar itens e considerar cortes guilhotinados. Outras características que podem ser incorporadas dizem respeito às incertezas, que podem estar relacionadas aos itens e recipientes (ALEM *et al.*, 2010; CRAINIC *et al.*, 2014).

Esta tese está relacionada com o estudo de problemas de corte de itens irregulares, também conhecidos como problemas de *nesting* (OLIVEIRA, 1995), em particular problemas com duas dimensões. No levantamento de Wäscher, Haussner e Schumann (2007), dos 445 trabalhos publicados sobre problemas de corte e empacotamento, 64 deles lidaram com problemas envolvendo itens irregulares. Em um levantamento feito recentemente sobre modelos matemáticos para

problemas de corte de itens irregulares, Leão *et al.* (2020) encontraram 27 trabalhos, ressaltando a carência de pesquisas na área e também que há pouca ou nenhuma inclusão de características ou restrições que surgem em contextos reais quando se resolve esses problemas. Como observado por Mundim (2017), o número de trabalhos sobre problemas de corte de itens irregulares é ainda menor quando se considera a presença de incertezas nos parâmetros.

O interesse particular desta tese está em problemas de *nesting* do tipo mochila (ou seja, *single knapsack problem* na tipologia de Wäscher, Haussner e Schumann (2007)) e de uma dimensão aberta (isto é, o problema de corte em faixa) (ou seja, *open dimension problem* na tipologia de Wäscher, Haussner e Schumann (2007)). Objetiva-se contribuir com a literatura desses problemas a partir do desenvolvimento de métodos de solução e a modelagem de incertezas. Em particular, propõe-se modelos de programação matemática, resolvidos com métodos exatos, considerando a presença de incertezas. Busca-se também contribuir com os problemas sem incertezas, com o desenvolvimento de heurísticas que sejam competitivas com aquelas da literatura. O número de contribuições para problemas do tipo mochila ainda é limitado (MARTINS; TSUZUKI, 2010; VALLE *et al.*, 2012; DALALAH; KHRAIS; BATAINEH, 2014; MUNDIM *et al.*, 2018; ROMÁN, 2020), com bem menos trabalhos em comparação com os problemas de corte em faixa (OLIVEIRA; GOMES; FERREIRA, 2000; ELKERAN, 2013; CHERRI *et al.*, 2016; LEÃO *et al.*, 2020). Na presença de incertezas, conhece-se somente o estudo feito por Mundim (2017) para o problema de corte de estoque.

O desenvolvimento desta tese segue uma abordagem quantitativa, pois trabalha sobre números para obter as conclusões sobre os métodos de solução desenvolvidos. Considerando seus objetivos, a tese é classificada como descritiva, por identificar e propor métodos de solução que possam dar uma nova visão sobre os problemas em estudo. Além disso, a sua finalidade é aplicada, pois os métodos são desenvolvidos buscando um fim real, por exemplo, o que fazer quando as demandas de itens são incertas. Em termos de procedimento, parte-se de materiais já publicados, como artigos, livros e teses, com o fim de obter informações sobre a literatura para, então, identificar possibilidades de melhorias e trazer novas contribuições. Considera-se também a realização de experimentos computacionais sobre dados disponíveis em outros trabalhos da literatura, a fim de observar o comportamento e a eficácia dos métodos propostos (KOTHARI, 2004).

1.1 Aspectos geométricos

Os problemas de interesse dessa tese consideram um conjunto de itens, em que pelo menos um item tem formato irregular. Cada item é descrito por um conjunto de vértices ordenados, sendo um dos vértices tomado como o vértice de referência, que é por onde o item é posicionado no recipiente. A cada item está associada uma área, um valor, uma demanda e um conjunto de possíveis rotações e é caracterizado como um polígono convexo ou não convexo, com a

possibilidade de ter buracos. Caso o item possua alguma parte curva, faz-se uma aproximação poligonal (isto é, representa-se a curva por segmentos de retas consecutivos e definidos sobre a curva) tão precisa quanto se deseja. A Figura 1.1 exemplifica alguns tipos de itens.

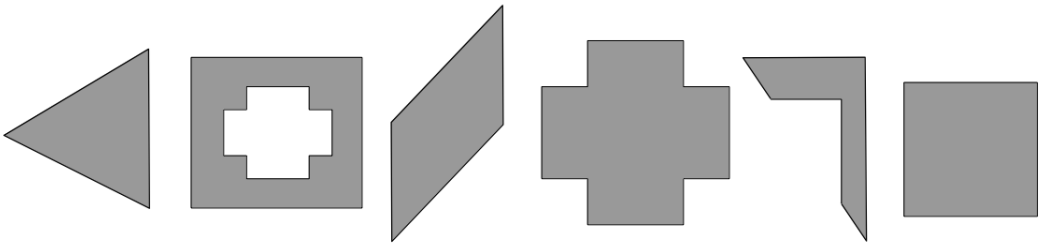


Figura 1.1 – Exemplo de itens regulares e irregulares.

Os problemas estudados são bidimensionais, com a representação ocorrendo no primeiro quadrante do Plano Cartesiano, com a origem do recipiente no ponto $(0,0)$. Associa-se o eixo x com a dimensão da largura e o eixo y com a dimensão do comprimento (ou altura). Assuma-se que no problema da mochila o recipiente é retangular, podendo ter buracos ou defeitos. Um recipiente com defeitos ou buracos pode ser modelado como um recipiente retangular, em que os defeitos ou buracos são interpretados como itens já empacotados (MUNDIM, 2015). No caso do problema com uma dimensão aberta, assume-se que o recipiente é retangular e tem uma das dimensões aberta (isto é, que precisa ser definida durante a resolução do problema). A Figura 1.2 mostra exemplos de recipiente para esses problemas.

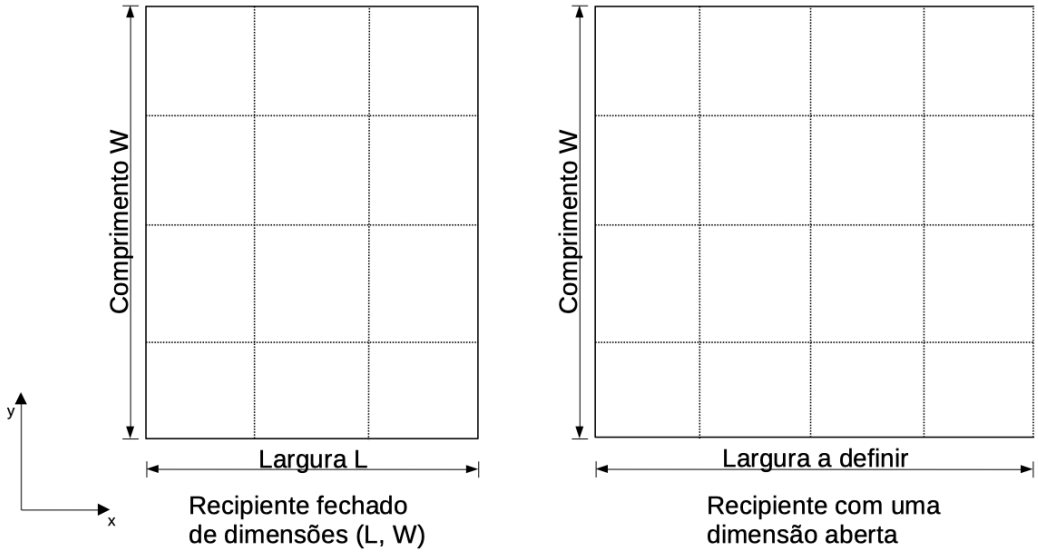


Figura 1.2 – Exemplo de recipiente fechado, para o problema da mochila, e com uma dimensão aberta, para o problema de corte em faixa.

Obter uma solução viável para problemas contendo itens irregulares exige lidar com a geometria dos itens. Um tutorial de como lidar com a representação e a geometria de itens irregulares foi feito por Bennell e Oliveira (2008). Os autores destacaram as seguintes ferramentas para garantir soluções sem sobreposição e respeitando as dimensões do recipiente: método *raster*,

phi-function, trigonometria direta e *no-fit polygon*. Outras ferramentas surgiram a partir dessas. Sato, Martins e Tsuzuki (2012) definiram uma região livre de colisão a partir do cálculo do *no-fit polygon* dos itens e suas interseções. Toledo *et al.* (2013) combinaram o *no-fit polygon* com o método *raster*, resultando no *no-fit raster*. Cherri *et al.* (2018) propuseram uma estrutura de dados para salvar as informações geométricas resultantes do cálculo do *no-fit raster*. Sato *et al.* (2019) partiram do *no-fit raster* para criar mapas de sobreposição, sendo definida uma função que indica quão sobrepostos estão dois itens.

No método *raster*, os itens e o recipiente são discretizados em um malha representada por uma matriz. A matriz dos itens contém o valor "um" em cada posição, enquanto a matriz do recipiente contém o valor "zero". Quando um item é posicionado no recipiente, as posições da matriz do recipiente cobertas pela matriz do item são atualizadas com a soma dos valores nas respectivas posições cobertas, mesmo que uma posição seja coberta parcialmente (SEGENREICH; BRAGA, 1986). Se alguma posição da matriz do recipiente contiver um valor maior do que "um", então a sobreposição ocorre. A vantagem em se utilizar o método *raster* está na forma rápida de detectar itens se sobrepondo ou extrapolando as dimensões do recipiente. Uma desvantagem deste método está relacionada com a precisão imposta para discretizar os itens e recipiente, influenciando diretamente no tamanho das matrizes (OLIVEIRA; FERREIRA, 1993), bem como pode não ocorrer encaixes perfeitos entre os itens.

Na trigonometria direta, verifica-se se existe a interseção entre segmentos de retas correspondentes às arestas dos itens e a inclusão de algum vértice de um item dentro de outro item. Para o cálculo da interseção entre segmentos, pode-se adotar a função D, que descreve a posição relativa de um ponto em relação a uma reta, que também pode ser utilizada para verificar a posição relativa entre dois segmentos (MAHADEVAN, 1984). No caso do teste de inclusão, pode-se calcular o número de voltas (*winding number*), baseado na soma dos ângulos, ou os cruzamentos de um raio (*ray crossings*), baseado no número de vezes que uma semi-reta horizontal que sai do vértice cruza os segmentos do outro item (KUMAR; BANGI, 2018). Rocha *et al.* (2014) e Rocha *et al.* (2016) usaram círculos para representaram os itens e assim verificar a não sobreposição entre eles. Por outro lado, Peralta, Andretta e Oliveira (2018) propuseram usar retas de separação definidas a partir de dois vértices consecutivos de um item. Dois itens que não se sobrepõem vão possuir seus vértices em lados diferentes da reta ou sobre a reta. Uma desvantagem da trigonometria direta está relacionada com a necessidade de refazer os cálculos sempre que algum item mudar de posição dentro do recipiente.

O *no-fit polygon* considera a construção de um polígono de obstrução entre pares de itens (ALBANO; SAPUPPO, 1980). Um dos itens é considerado fixo e o outro é definido como orbital. O item orbital é posicionado de tal forma a encostar no item fixo. A partir de um ponto de referência do item orbital, translada-se esse ao redor do item fixo, sempre os deixando encostados e sem que ocorra qualquer tipo de sobreposição. O caminho percorrido pelo ponto de referência define o *no-fit polygon*, de forma que se o item orbital tiver o seu ponto de referência

sempre posicionado sobre o *no-fit polygon*, então os itens estão encostados. Caso o ponto de referência esteja no exterior do *no-fit polygon*, os itens estão separados. Caso contrário, os itens estão se sobrepondo. A vantagem do *no-fit polygon* é que ele pode ser obtido em uma etapa de pré-processamento. Burke *et al.* (2007) forneceram uma descrição detalhada de como calcular o *no-fit polygon* entre itens com buracos e não convexos. Burke *et al.* (2010) generalizaram a obtenção do *no-fit polygon* para itens com arcos e curvas, cobrindo todos os casos possíveis de tipos de itens. Porém, o algoritmo apresentado por esses autores parece ser demasiadamente complexo para ser implementado, devido aos vários casos que podem surgir, por exemplo, ao ter muitas rotações para os itens.

Outra forma de calcular o *no-fit polygon* é pelo método de CUNINGHAME-GREEN (1989), definido para itens convexos. Esse método considera que o item fixo está definido sobre uma orientação anti-horária e o item orbital sobre uma orientação horária. O vértice de referência do polígono orbital corresponde àquele de maior ordenada (em caso de empate, o de maior abscissa). Assim, transladam-se todos os segmentos de reta do item fixo e orbital para o vértice do item fixo de menor ordenada (em caso de empate, o de menor abscissa). Por fim, os segmentos de retas são concatenados no sentido anti-horário para obter o *no-fit polygon*. No caso de algum item ser não-convexo, uma estratégia é fazer a sua divisão em partes convexas (por exemplo, usando algum procedimento de triangulação), tal que se calcula o *no-fit polygon* entre todos os pares de partes convexas de cada item. Ao final, faz-se a união dos *no-fit polygons* das partes para gerar o *no-fit polygon* final entre os itens. Neste caso, obter o *no-fit polygon* pode se tornar demasiadamente pesado conforme a geometria dos itens torna-se complexa (MUNDIM, 2015).

Outra ferramenta que pode ser derivada das mencionadas anteriormente consiste na combinação do *no-fit polygon* com o método *raster*, conforme descrito em Toledo *et al.* (2013), resultando no *no-fit raster*. Neste caso, a cada *no-fit polygon* se associa uma matriz cujas posições com o valor "um" indicam que os itens estão se sobrepondo, caso contrário, tem-se o valor "zero". A vantagem do *no-fit raster* é que a sobreposição pode ser verificada percorrendo uma matriz que representa a discretização do *no-fit polygon*, ao invés de percorrer a representação real do *no-fit polygon*. Além disso, em comparação com o método *raster*, o *no-fit raster* pode permitir o encaixe perfeito entre itens, característica herdada do *no-fit polygon*. Na Figura 1.3, tem-se o exemplo de um *no-fit raster* para um triângulo, que é o item orbital, é um retângulo, que é o item fixo, resultando na matriz de uns e zeros.

A *phi-function* considera funções matemáticas que relacionam pares de itens e podem ser vistas como uma generalização do *no-fit polygon*. Essas funções indicam a posição de um item com relação ao outro (STOYAN *et al.*, 2001; STOYAN *et al.*, 2004). Quando a função resulta no valor "zero", então os itens estão se encostando; quando o resultado é maior do que zero, os itens estão separados; caso contrário, os itens estão se sobrepondo. Stoyan *et al.* (2001) definiram funções para relacionar itens de geometria mais simples, como retângulos e círculos, além de indicarem que funções para itens mais complexos podem ser obtidas fazendo a composição de

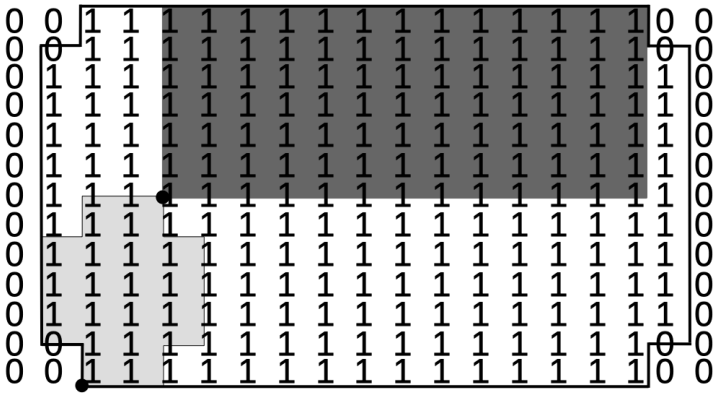


Figura 1.3 – *No-fit raster* de um retângulo (item fixo) e um item na forma de cruz (item orbital).

funções elementares. Chernov *et al.* (2012) apresentaram uma metodologia para obter funções quando há itens representados por linhas e arcos. Embora a *phi-function* permita verificar a sobreposição entre quaisquer tipos de itens, ela ainda tem sido pouca adotada na literatura pela dificuldade em definir funções para itens com geometrias complexas (por exemplo, não convexos ou com buracos). Visando suprir essa dificuldade, Stoyan *et al.* (2015) e Stoyan, Pankratov e Romanova (2016) propuseram as *quasi phi-function*, uma generalização da *phi-function* com a inclusão de variáveis auxiliares.

As ferramentas anteriores são utilizadas, em especial, para detectar a sobreposição entre itens. No caso de verificar se os itens estão contidos dentro do recipiente, um conceito utilizado é o de *inner-fit polygon*, calculado para cada item e o recipiente. O *inner-fit polygon* é o polígono obtido ao transladar o item pelo seu vértice de referência, de maneira que o item esteja sempre encostando em pelo menos uma das bordas do recipiente. Ele contém as posições que um item pode ser posicionado de forma a ficar inteiramente contido no recipiente (GOMES; OLIVEIRA, 2002). Pode-se discretizar o *inner-fit polygon* sobre uma matriz de pontos, associando o valor "zero" para uma posição válida para posicionar o item e, caso contrário, adota-se o valor "um". A Figura 1.4 ilustra o *inner-fit raster* para um item e um recipiente retangular.

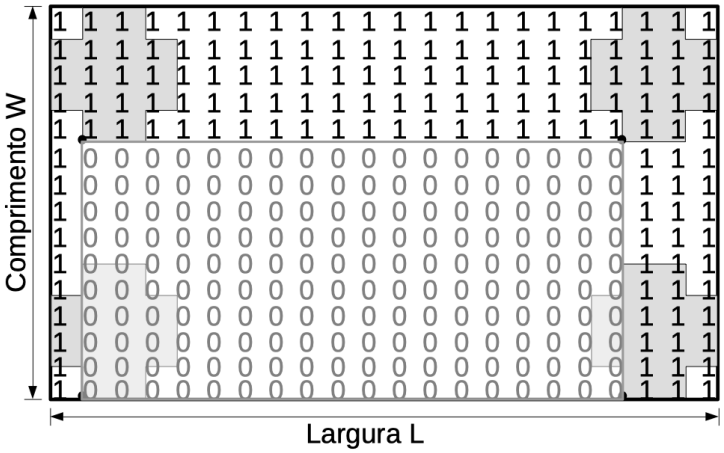


Figura 1.4 – *Inner-fit raster* de um item na forma de cruz.

1.2 Revisão da literatura

A grande maioria da literatura de problemas de *nesting* considera o emprego de heurísticas, por se tratarem de problemas NP-Difíceis (FOWLER; PATERSON; TANIMOTO, 1981), embora existam métodos exatos, relacionados principalmente com a resolução de modelos de programação linear inteira. Apresenta-se adiante uma revisão das principais contribuições envolvendo os problemas bidimensionais que são de interesse para esta tese. Ressalta-se que várias dissertações e teses, como Oliveira (1995), Sykora (2013), Silveira (2013), Cherri (2016), Aureliano (2017), Silva (2017), Abeysooriya (2017), Polo (2018), trazem revisões de métodos exatos e heurísticas para diferentes problemas de *nesting*.

Com relação ao problema da mochila, Scheithauer e Terno (1993) apresentaram modelos de programação linear inteira mista em que as restrições de não sobreposição são derivadas a partir do *no-fit polygon*. Martins e Tsuzuki (2010) propuseram uma heurística baseada em recozimento simulado para o empacotamento em mochilas retangulares e irregulares assumindo que os itens irregulares possuem rotação livre. Os autores calculam uma região livre de colisão usando o conceito de *no-fit polygon* para determinar o local de empacotar os itens. Valle *et al.* (2012) apresentaram heurísticas baseadas em GRASP (*Greedy Randomized Adaptive Search Procedure*) . Os autores resolveram duas variações desse problema: mochila 0-1, em que cada item pode ser empacotado apenas uma vez; e, mochila ilimitada, em que não existe um limite no número de cópias de um item que podem ser empacotadas. Para empacotar um item, verificam-se as posições viáveis a partir do *no-fit polygon* do item com os itens já empacotados. Mundim e Queiroz (2012) fizeram a combinação do GRASP de Valle *et al.* (2012) com o recozimento simulado, sendo que este último é utilizado na fase de busca local do GRASP para diversificar a busca e escapar de ótimos locais.

No problema da mochila, Silveira (2013) desenvolveu um algoritmo genético, em que cada cromossomo representa uma sequência para empacotar os itens. Os itens são empacotados por uma heurística de agrupamento, que olha o *no-fit polygon* dos itens para determinar a melhor posição para empacotar. Uma busca local, que considera três operadores, é utilizada visando mudar a sequência dos itens e explorar novas soluções. Dalalah, Khrais e Bataineh (2014) consideraram recipientes retangulares e irregulares, propondo uma heurística construtiva, que empacota os itens conforme a utilização da envoltória convexa. Baldacci *et al.* (2014) consideraram o recipiente irregular, com aplicação na indústria de couro. Os autores discretizaram o recipiente sobre uma malha, apresentando um modelo linear inteiro e uma relaxação lagrangeana. Os autores também apresentaram heurísticas a partir da relaxação proposta, que consideram formas distintas para empacotar os itens. Mundim *et al.* (2018) desenvolveram uma heurística geral que usa diferentes regras de posicionamento, algumas delas baseadas na *bottom-left*. Os autores conseguiram melhorar grande parte das soluções apresentadas por trabalhos anteriores. Recentemente, Cherri *et al.* (2019) desenvolveram modelos de programação por restrições que permitem resolver diferentes problemas de *nesting*, incluindo o problema da mochila. Os autores

propuseram uma restrição global para garantir que itens não se sobreponham.

Há um maior número de trabalhos para o problema de corte em faixa. Com relação à proposta de modelos matemáticos, Li e Milenkovic (1995) desenvolveram um modelo de programação linear que considera a posição relativa dos itens e que os itens sejam deslocados todos ao mesmo tempo para reduzir a dimensão variável do recipiente. Em Carravilla *et al.* (2003), há modelos resolvidos por programação por restrições, que consideram itens convexos e não convexos, em que a não sobreposição é verificada pelo *no-fit polygon* e há restrições para remover soluções simétricas. Em Fischetti e Luzzi (2009), há um modelo de programação linear inteira mista que define a posição dos itens por meio de variáveis contínuas. Para gerar as restrições de não sobreposição, os autores usaram variáveis binárias que relacionam partições do *no-fit polygon*. Com isso, dois itens não se sobrepõem se o ponto de referência de um deles está em uma das partições do *no-fit polygon*. Esse modelo foi melhorado por Alvarez-Valdes, Martinez e Tamarit (2013), que mostraram como realizar partições do *no-fit polygon* quando ele é não convexo e melhorar os limitantes das variáveis contínuas por meio de uma técnica de *lifting*. Os autores implementaram o seu próprio algoritmo *branch-and-bound*, testando diferentes formas de realizar a ramificação e métodos para fixar variáveis. Toledo *et al.* (2013) desenvolveram um modelo de programação linear inteira mista que considera o recipiente discretizado sobre uma malha. As variáveis de decisão associam o posicionamento do ponto de referência dos itens sobre pontos da malha, com os domínios estabelecidos pelo *inner-fit raster*. As restrições para evitar a sobreposição de itens foram derivadas do *no-fit raster*.

Ainda considerando modelos para o problema de corte em faixa, o modelo linear inteiro misto de Leão *et al.* (2016) considerou variáveis semi-contínuas, com um dos eixos sendo discretizado (eixo y). Esse modelo considera o *inner-fit polygon* para determinar o domínio das variáveis e o *no-fit polygon* para determinar as restrições de não sobreposição, sendo competitivo com o modelo de Toledo *et al.* (2013). Cherri *et al.* (2016) propuseram dois modelos, que podem lidar com a rotação e itens com buracos. O primeiro modelo adota a trigonometria direta para modelar as restrições de sobreposição entre itens, enquanto o segundo modelo usa o *no-fit polygon* para evitar a sobreposição e tem algumas desigualdades válidas definidas. Os autores fizeram vários experimentos computacionais e chegaram na conclusão de que o segundo modelo é melhor do que os anteriores apresentados na literatura. Rodrigues e Toledo (2017) apresentaram melhorias sobre o modelo de Toledo *et al.* (2013), desenvolvendo restrições de clique para lidar com a não sobreposição. As novas restrições são obtidas sobre cliques de um grafo de conflitos, em que os vértices representam as variáveis binárias do modelo e as arestas indicam que os vértices se sobrepõem. Os autores consideraram heurísticas da literatura para obter as cliques e montar as restrições de não sobreposição. Em Peralta, Andretta e Oliveira (2018) há um modelo de programação não linear que considera rotação livre para os itens, sendo os itens não convexos divididos em convexos, de forma que as restrições de não sobreposição são obtidas por trigonometria direta através de retas de separação. Essas retas têm o objetivo de separar as partes convexas dos itens e são modeladas a partir de uma equação da reta no plano.

Os autores enfatizaram que o seu modelo é capaz de ter menos variáveis. Recentemente, Leão *et al.* (2020) revisaram os vários modelos matemáticos propostos para problemas de *nesting*, incluindo o de corte em faixa. Os autores apontaram contribuições de 12 trabalhos sobre modelos de programação linear inteira, 12 trabalhos que propuseram modelos de programação não linear e 4 trabalhos que lideram com modelos de programação por restrições.

Com relação a heurísticas desenvolvidas para o problema com uma dimensão aberta, Baker, Coffman Jr. e Rivest (1980) empregaram a *bottom-left*, que, a partir de uma sequência para empacotar os itens, iterativamente seleciona um item da sequência e posiciona-o mais à esquerda e abaixo possível dentro do recipiente. Albano e Sapuppo (1980) buscavam, dentro da sequência de itens, o melhor deles que poderia ser empacotado por uma heurística *bottom-left* na iteração corrente. Jakobs (1996) desenvolveu um algoritmo genético em que cada indivíduo representa uma sequência para empacotar os itens. Os itens são empacotados por uma heurística *bottom-left*, posicionando inicialmente suas envoltórias retangulares para, em seguida, aplicar um método de compactação para reduzir o tamanho da dimensão variável do recipiente. Dowsland, Dowsland e Bennell (1998) propuseram variações da heurística *bottom-left*, empacotando itens tanto do lado direito quanto do lado esquerdo do recipiente e com algumas heurísticas aproveitando os buracos que surgem durante o empacotamento para posicionar itens. Para lidar com a sobreposição entre itens, os autores empregaram a trigonometria direta e o *no-fit polygon*.

Oliveira, Gomes e Ferreira (2000) desenvolveram um algoritmo construtivo, chamado TOPOS (Técnicas de Optimização para o Posicionamento de Figuras Irregulares) , que usa critérios como comprimento, área, complexidade dos itens e envoltória retangular, para selecionar os itens a serem empacotados. Soluções parciais eram consideradas como um único item, de forma que o *no-fit polygon* foi usado para determinar a melhor posição para empacotar os próximos itens. Gomes e Oliveira (2002) trabalharam sobre uma sequência de itens, com as posição iniciais válidas para empacotar os itens sendo obtidas do *inner-fit polygon* e depois considerando os vértices e pontos sobre o *no-fit polygon*. Critérios foram adotados, seguindo Oliveira, Gomes e Ferreira (2000), para gerar sequências para empacotar os itens por uma heurística *bottom-left*. Uma busca local foi considerada para trocar itens de posição nas sequências, sendo que a quantidade de trocas é controlada por um parâmetro. Em Gomes e Oliveira (2006), há um recozimento simulado, com a heurística *bottom-left* para posicionar os itens, um modelo de programação linear para remover sobreposições e outro modelo de programação linear para realizar a compactação da solução. O *inner-fit polygon* e *no-fit polygon* foram empregados para atender as restrição do problema, enquanto novas soluções são geradas pelo recozimento simulado, trocando ou movendo itens de lugar na sequência.

Ainda com relação a heurísticas para o problema com uma dimensão aberta, Burke *et al.* (2006) consideraram a *bottom-left* como heurística para empacotar os itens sobre uma malha semi-contínua que representa o recipiente, tentando preencher os buracos que surgem no empacotamento e utilizando o *no-fit polygon* para evitar a sobreposição entre itens. Sequências de

itens são geradas por uma heurística *hill climbing*, que realiza a troca de itens de posição, e uma busca tabu, que evita gerar sequências repetidas, armazenando-as em uma lista tabu. Imamichi, Yagiura e Nagamochi (2009) desenvolveram uma heurística de busca local iterada, com um procedimento de separação baseado em programação não linear e um procedimento de troca de itens. Wong *et al.* (2009) propuseram um algoritmo genético para determinar a sequência em que os itens devem ser empacotados e uma heurística de dois níveis para posicionar os itens sobre uma malha. Elkeran (2013) desenvolveu uma heurística híbrida que combina *cuckoo search* e busca local guiada. O autor considerou o agrupamento de itens em uma fase de pré-processamento, enquanto que a heurística utiliza o *no-fit polygon* para lidar com a sobreposição entre itens. Ao obter uma solução viável, a heurística busca reduzir o tamanho da dimensão variável por um método de compactação, o qual pode ocasionar uma solução com sobreposição. Em seguida, outro método foi utilizado para reduzir ou eliminar a sobreposição que pode existir na solução.

Em Pinheiro, Amaro Jr. e Saraiva (2016), há um algoritmo genético de chaves aleatórias que usa heurísticas baseadas em *bottom-left* para posicionar os itens. Os cromossomos contêm informações da sequência e da rotação que os itens devem ser posicionados. Mundim, Andretta e Queiroz (2017), por outro lado, desenvolveram um algoritmo genético de chaves aleatórias viciadas, que utiliza a heurística *bottom-left* para o posicionamento de itens. Nesse algoritmo, o cromossomo indica a sequência e a rotação para empacotar os itens. Os autores consideraram duas versões, uma com probabilidade viciada para o filho herdar informações do pai elite e a outra versão sem a probabilidade viciada, sendo que a versão com probabilidade viciada teve desempenho melhor. Por sua vez, Amaro Jr., Pinheiro e Coelho (2017) desenvolveram uma versão paralela e com múltiplas populações do algoritmo genético de chaves aleatórias viciadas. Os autores usaram o conceito de região livre de colisão para evitar a não sobreposição entre itens. Sato *et al.* (2019) propuseram uma heurística que busca pela minimização da sobreposição entre itens. Os autores definiram uma função que indica quão sobrepostos estão dois itens, usando algoritmos para separar itens se sobrepondo, como uma busca local guiada.

Outros problemas de *nesting* também resolvidos na literatura são o com duas dimensões abertas, de empacotamento em recipientes e o de corte de estoque. Para o problema com duas dimensões abertas, o número de trabalhos na literatura é limitado. Em Stoyan e Patsuk (2000), considera-se apenas um tipo de item de forma que os autores usaram a *phi-function* para evitar a sobreposição entre itens. Em Birgin e Sobral (2008), há modelos não lineares para determinar o menor recipiente (que pode ser um retângulo, círculo ou triângulo) dado o empacotamento de um conjunto de círculos. Considerando o empacotamento de círculos e polígonos convexos, com rotações livres para os itens, Kallrath (2009) desenvolveu modelos de programação não linear para determinar o menor quadrado capaz de conter todos os itens. Bennell *et al.* (2015) também desenvolveram modelos não lineares, com foco em determinar o melhor recipiente em termos de mínima área, perímetro ou um dado coeficiente, considerando o agrupamento de itens. O recipiente pode ser um retângulo, círculo ou polígono convexo, enquanto os itens podem

conter segmentos de reta ou curvas, sendo permitida a rotação livre. Por envolver apenas pares de itens, para verificar a sobreposição, os autores utilizaram a *phi-function*. Em Mundim, Andretta e Queiroz (2017), o algoritmo genético de chaves aleatórias viciadas também foi aplicado para resolver o problema com duas dimensões abertas, embora os autores tenham apresentado um modelo de programação não linear inteira mista, mas que não foi utilizado nos experimentos. O algoritmo se mostrou bastante competitivo quando aplicado sobre as instâncias geradas, bem como melhorou os resultados de instâncias pequenas da literatura.

Com relação às contribuições para o problema de empacotamento em recipientes, Terashima-Marín *et al.* (2010) propuseram uma hiper-heurística baseada em algoritmos genéticos. Os autores criaram procedimentos para selecionar itens e recipientes e depois para posicionar os itens nos recipientes. A hiper-heurística de López-Camacho *et al.* (2014) também é baseada em algoritmos genéticos e busca pela seleção e aplicação de heurísticas simples com base nas instâncias e problema sendo resolvido. Mundim *et al.* (2018) melhoraram os resultados anteriores da literatura com a sua heurística geral. Sykora *et al.* (2017) resolveram um problema de uma indústria cerâmica, permitindo rotação livre para os itens. Além disso, os autores propuseram heurísticas construtivas que aproveitam modelos de programação linear inteira para combinar itens e posicionar itens nos recipientes. Abeysooriya, Bennell e Sykora (2018) também permitiram que os itens tivessem rotação livre e então desenvolveram um algoritmo construtivo que usa um outro procedimento para decidir o posicionamento de itens. Por sua vez, Bennell, Cabo e Sykora (2018) consideraram a restrição de corte guilhotinado, ou seja, cortes que vão em linha reta de um lado ao outro do recipiente, no problema de empacotamento em recipientes de tamanhos variados. Os autores apresentaram um algoritmo *beam search* que trabalha sobre uma árvore de busca.

Para o problema de corte de estoque, Valle *et al.* (2012) apresentaram uma heurística baseada em geração de colunas. Primeiro, os itens são combinados para obter retângulos com uma alta taxa de ocupação. Em seguida, um algoritmo de programação dinâmica posiciona os retângulos para gerar colunas válidas. Song e Bennell (2014a) apresentaram heurísticas baseadas na geração de colunas e procedimentos sequenciais. Uma heurística do tipo *beam search* foi usada para gerar as colunas. Xu (2016) apresentou heurísticas baseadas em encaixe e na *bottom-left* para um problema da indústria naval, em que os itens são posicionados em uma malha calculada dos *no-fit polygons*. Mundim *et al.* (2018) aplicaram a sua heurística geral sobre o problema de corte de estoque, obtendo muitas soluções melhores do que Valle *et al.* (2012) e Song e Bennell (2014a).

Há também trabalhos com métodos direcionados para aplicações na indústria de couro, automobilística e metal-mecânica, considerando recipientes irregulares e com zonas de qualidade e/ou defeitos. Heistermann e Lengauer (1995) resolveram um caso real encontrado na indústria de couro por meio de uma heurística que busca pela posição mais adequada para empacotar um item, de uma lista de itens candidatos, em recipientes irregulares com zonas de qualidade

e defeitos. Crispin *et al.* (2005) lidaram com um problema da indústria de calçados, propondo algoritmos genéticos. Lee, Ma e Cheng (2008) resolveram um problema que considera múltiplos recipientes irregulares e com defeitos. Alves *et al.* (2012a) e Alves *et al.* (2012b) lidaram com um problema da indústria automobilística, propondo uma heurística de busca em vizinhança variável que considera a busca sobre a sequência de itens. Por sua vez, Pinto *et al.* (2016) melhoraram os resultados de Alves *et al.* (2012b), aplicando uma heurística construtiva que simula o empacotamento de itens em pontos do recipiente. Chen *et al.* (2020) lidaram com o problema de corte de ardósias, que considera o recipiente irregular e com defeitos. Os autores desenvolveram uma heurística que divide o recipiente em níveis para o posicionamento dos itens.

1.3 Problemas sob incertezas

Com relação a considerar incertezas em problemas de *nesting*, conhece-se somente o trabalho de Mundim (2017). O autor considerou a demanda dos itens incerta, para o problema de corte de estoque, com a proposta de um modelo de programação estocástica de dois estágios com recurso. A função objetivo do modelo penaliza a falta ou o excesso de itens em comparação com a demanda dada por um conjunto de cenários amostrados. O autor usou a heurística geral de Mundim *et al.* (2018) para obter soluções viáveis para um algoritmo de geração de colunas.

De acordo com Birge e Louveaux (1997), dentro da programação estocástica, os parâmetros incertos (aleatórios) de um problema de otimização podem ser modelados através de variáveis aleatórias. Essas variáveis são representadas por algum espaço de probabilidade, que inclui o conjunto de possíveis estados (eventos). Cada estado representa o resultado da realização (ocorrência) das variáveis aleatórias. Ao associar uma distribuição de probabilidade discreta para os parâmetros incertos, cria-se um número finito de cenários que representam as realizações das variáveis aleatórias (que modelam os parâmetros incertos). Cada cenário tem uma probabilidade de ocorrência, que simboliza a chance do estado de fato se materializar.

Em um modelo de programação estocástica de dois estágios com recurso, ações de correção podem ser tomadas após a realização das variáveis aleatórias. As variáveis (decisões) de primeiro estágio são determinadas inicialmente, em seguida, tem-se a realização das variáveis aleatórias para, então, determinar as variáveis de segundo estágio (isto é, decisões de recurso que buscam corrigir, se preciso, as decisões de primeiro estágio). A ideia é que a solução desse modelo seja equilibrada diante da ocorrência de todos os possíveis estados. Dessa forma, as decisões de primeiro estágio são definidas levando em consideração a função de recurso que modela o segundo estágio. A vantagem em se usar uma abordagem por cenários é que o modelo de programação estocástica de dois estágios com recurso se transforma em um modelo determinístico (KALL; WALLACE, 1994).

Além de discutir sobre programação estocástica como uma metodologia para modelar incertezas de parâmetros em problemas de otimização, Alem e Morabito (2015b) comentaram

sobre a otimização robusta. Segundo os autores, a programação estocástica assume que é conhecida a distribuição de probabilidade associada aos parâmetros incertos, ao passo que a otimização robusta não necessita conhecer essa distribuição. Na programação estocástica, que está associada com a otimização do valor esperado, é possível incorporar diferentes medidas de aversão ao risco, embora seja uma metodologia limitada pela quantidade de cenários (KALL; WALLACE, 1994). Por outro lado, na otimização robusta, que assegura uma análise de pior caso, não é preciso considerar cenários, mas é preciso trabalhar sobre soluções ótimas (BERTSIMAS; SIM, 2003). Outras metodologias incluem a programação dinâmica, que também requer o conhecimento da distribuição de probabilidade (BELLMAN, 1957), e a programação *fuzzy*, que modela os parâmetros como números *fuzzy* (BELLMAN; ZADEH, 1970).

Existem algumas contribuições na literatura para quando os itens são regulares ou se considera o problema com apenas uma dimensão (por exemplo, o volume). Beraldi, Bruni e Conforti (2009) resolveram o problema de corte de estoque com incertezas na demanda, avaliando o impacto no lucro da baixa ou superprodução dos itens. Perboli, Tadei e Baldi (2012) resolveram o problema de empacotamento em recipientes generalizado, em que os itens possuem volume e valor, enquanto os recipientes têm custo e volume máximo de capacidade. Nesse problema, o valor dos itens é incerto, a depender das operações de manuseio que os recipientes recebem.

Em Crainic *et al.* (2016), uma variante do problema de empacotamento em recipientes generalizado foi resolvida, considerando que a demanda dos itens é um parâmetro incerto, dada a incerteza sobre as necessidades futuras por itens e os custos associados às atividades de manuseio. Os autores propuseram um modelo de programação estocástica de dois estágios, além de uma nova meta-heurística baseada em *progressive hedge* para acelerar as buscas por uma boa solução. Recentemente, Liu, Deng e Li (2019) resolveram um problema de empacotamento de itens em recipientes e empilhamento de recipientes dentro de armazéns. Os autores trataram a demanda dos itens como um parâmetro incerto e propuseram um modelo de dois estágios que utiliza cenários e outro que utiliza intervalos para a realização da demanda. Os resultados computacionais indicaram que a solução do problema combinado traz benefícios econômicos e de melhor utilização dos espaços dos armazéns.

Outro problema que tem estudos considerando incertezas é o problema da mochila, em especial, na sua versão unidimensional. Monaci, Pferschy e Serafini (2013) assumiram que o volume dos itens não é exatamente conhecido *a priori*, mas está definido em um intervalo. Os autores desenvolveram um algoritmo de programação dinâmica, além de técnicas para diminuir a complexidade de espaço e tempo. Em Rooderkerk e Heerde (2016), busca-se otimizar a variedade de itens para diferentes lojas por meio da resolução do problema da mochila binária. Os autores consideraram uma medida de risco associada às variedades de itens, além de assumir que a demanda e o lucro por item são parâmetros incertos. A resolução do problema ocorreu por meio de uma heurística para construir a fronteira entre retorno e risco. Range, Kozlowski e Petersen

(2018) resolveram o problema da mochila considerando que o volume dos itens não é conhecido exatamente. Os autores utilizaram restrições que permitiam violar a capacidade da mochila dentro de um certa probabilidade, além de adicionarem uma penalidade na função objetivo. A resolução do problema se deu com um algoritmo de programação dinâmica para o problema do caminho mais curto com restrições de recurso.

1.4 Contribuições e organização da tese

O Capítulo 1 da tese introduz os problemas de corte e empacotamento, suas aplicações, a tipologia proposta na literatura e as classificações existentes. Apresenta-se brevemente os problemas de interesse, que consideram itens irregulares e com uma dificuldade maior relacionada à geometria dos itens. Descrevem-se as principais ferramentas geométricas para lidar com a sobreposição de itens e também permitir que itens sejam posicionados inteiramente dentro de recipientes. Em seguida, apresenta-se uma revisão da literatura, fazendo uma discussão geral das contribuições dos principais trabalhos que resolveram os problemas do tipo mochila e de corte em faixa, que são de interesse dessa tese, bem como de trabalhos sobre outros problemas de *nesting*, sem e com incertezas, que a literatura tem resolvido.

As contribuições dessa tese dizem respeito ao estudo e desenvolvimento de métodos para problemas de *nesting*, em particular, o problema da mochila e o problema de corte em faixa. Observando que o número de contribuições para o problema da mochila é relativamente menor em comparação ao problema de corte em faixa, contribui-se com heurísticas para o problema da mochila.

O Capítulo 2 traz as heurísticas propostas para o problema da mochila. A primeira é uma heurística baseada no algoritmo genético de chaves aleatórias viciadas, enquanto a segunda implementa uma busca em vizinhança variável. As heurísticas consideram formas diferentes de conduzir o processo de otimização, a primeira sobre múltiplas trajetórias (isto é, uma população de indivíduos) e a segunda considerando uma trajetória única. Propõe-se na implementação da primeira heurística que a probabilidade viciada de herdar informações dos melhores indivíduos faz parte do próprio indivíduo, ao invés de ser um parâmetro constante durante a otimização. Adota-se para as heurísticas que uma solução do problema é codificada por um vetor de inteiros. Ao invés do vetor conter apenas os números que identificam os itens, propõe-se incluir outros números para auxiliar durante o posicionamento dos itens. Dessa forma, não necessariamente um item vai ser posicionado na primeira posição válida do recipiente, ajudando a escapar de possíveis ótimos locais. O posicionamento de itens no recipiente segue a sequência obtida do vetor que representa a solução, adotando-se três regras, uma que é a *bottom-left* e outras duas com base nela. As duas heurísticas são comparadas com a heurística geral de Mundim *et al.* (2018), até então estado-da-arte para o problema da mochila, conseguindo obter resultados iguais ou melhores para todas as instâncias. A primeira heurística é capaz de obter soluções cuja ocupação

do recipiente é quase 7% maior, na média, enquanto a segunda heurística traz uma melhora na ocupação do recipiente de quase 6%, na média, comparado com Mundim *et al.* (2018). Para uma instância é possível melhorar a ocupação em quase 18%.

Parte dos resultados do Capítulo 2, envolvendo uma versão preliminar do algoritmo genético de chaves aleatórias viciadas, foi publicada como um artigo completo[1] e apresentada de forma oral na *XLIV Conferência Latino-americana de Informática - CLEI*, realizada entre os dias 01 e 05 de outubro de 2018 (SOUZA QUEIROZ; MUNDIM; ANDRETTA, 2018). Esse artigo foi escrito em colaboração com Marina Andretta (ICMC-USP) e Leandro Resende Mundim (ODM). O capítulo em si, com todos os resultados apresentados, foi publicado como artigo[2] no periódico *Applied Soft Computing* (SOUZA QUEIROZ; ANDRETTA, 2020b). Esse artigo foi escrito em colaboração com Marina Andretta (ICMC-USP).

Como na literatura de problemas de *nesting* não se conhece qualquer trabalho que lida com incertezas, além da contribuição de Mundim (2017), as demais contribuições da tese estão relacionadas a modelagem de incertezas em problemas da mochila e de corte em faixa.

No Capítulo 3, as incertezas são estudadas dentro do problema de corte em faixa considerando que não se conhece com precisão a demanda dos itens. Busca-se decidir qual o tamanho de área de faixa retangular a comprar. A área da faixa é determinada pelos itens que vão ser cortados. A demanda dos itens não é conhecida inicialmente, sendo revelada em um momento futuro. A faixa requer um determinado tempo de preparo pelo fornecedor, de forma que o preço por metro quadrado varia conforme o momento (inicial ou futuro) em que a área da faixa é encomendada. Se mais área de faixa é solicitada no futuro, a metragem adicional custa mais caro do que quando solicitada no momento inicial. Esse tipo de situação pode ocorrer na indústria têxtil, em que se busca cortar uma faixa (por exemplo, de tecido) para obter itens. Por exemplo, uma faixa de tecido é preparada observando questões de cor, tipo de material ou o número de fios em conformidade com os itens que se deseja obter (por exemplo, para a produção de roupas, bolsas ou sapatos). A indústria define a dimensão aberta da faixa para o fornecedor mediante os itens que ela precisa obter. A demanda de itens por clientes pode surgir em diferentes momentos. Se a indústria conhece toda a demanda com exatidão no momento mais cedo possível, ela consegue definir com o fornecedor a dimensão aberta da faixa e, assim, comprar a faixa no menor preço possível, já que o fornecedor tem um prazo maior para produzi-la. Por outro lado, quanto menor o prazo que o fornecedor tem para produzir a faixa, maior será o custo para a indústria. Dessa forma, mesmo a indústria não conhecendo a demanda exata dos itens no momento mais cedo possível, seria melhor definir com o fornecedor uma área inicial de faixa capaz de permitir (parcialmente ou por completo) a produção dos itens, sem ter que requisitar muito mais área adicional em um momento posterior, quando a indústria conhece todas as demandas.

Para o problema do Capítulo 3, desenvolve-se um modelo de programação estocástica

[1] https://ieeexplore.ieee.org/document/8786313
[2] https://www.sciencedirect.com/science/article/abs/pii/S1568494620304245

de dois estágios com recurso. O primeiro estágio tem sua função objetivo relacionada com o custo da área de faixa adquirida no momento mais cedo possível e os custos esperados pela aquisição adicional de área de faixa no momento futuro. No segundo estágio, busca-se definir o posicionamento dos itens na faixa conforme a realização de suas demandas, dada pelos diferentes cenários e casos de probabilidade, e assim verificar o quanto de área adicional seria preciso para cada cenário e caso. Para a resolução do modelo, propõe-se um algoritmo *branch-and-cut* que integra uma heurística baseada em busca em vizinhança variável para gerar soluções e limitantes válidos durante a resolução de nós da árvore de busca. A heurística é ainda adaptada para completar soluções parciais que advêm dos nós da árvore de busca. O algoritmo é competitivo com outros algoritmos exatos para o problema de corte em faixa sem incertezas, além de ter permitido a resolução do problema com incertezas contendo até 80 cenários. As análises do valor esperado da informação perfeita e do valor da solução estocástica indicaram que as incertezas impactam no problema e, por isso, a resolução do modelo de programação estocástica é economicamente vantajosa para melhor definir a área de faixa a comprar.

Alguns resultados do Capítulo 3, envolvendo o modelo de programação estocástica, foram publicados como artigo completo[3] e apresentados de forma oral no *LII Simpósio Brasileiro de Pesquisa Operacional - SBPO*, realizado entre os dias 03 e 05 de novembro de 2020 (SOUZA QUEIROZ; ANDRETTA, 2020a). Esse artigo foi escrito em colaboração com Marina Andretta (ICMC-USP). Todo o capítulo foi publicado como artigo[4] no periódico *International Transactions in Operational Research* (SOUZA QUEIROZ; ANDRETTA, 2022). Esse artigo foi escrito em colaboração com Marina Andretta (ICMC-USP).

No Capítulo 4, estuda-se o impacto de incertezas nos defeitos do recipiente para o problema da mochila. Parte-se da situação em que se precisa selecionar itens para produzir a partir do corte de um recipiente retangular com defeitos. Assume-se que o recipiente é comprado com antecendência, pois há questões de preparo e transporte por parte do fornecedor. As dimensões e características de material do recipiente são conhecidas, porém, os defeitos só são identificados após a entrega e quando o recipiente entra no estágio de corte. Isso significa que itens previamente selecionados para serem obtidos do recipiente podem ter a sua produção cancelada, incorrendo um custo de cancelamento. Esse tipo de situação pode aparecer, por exemplo, nas indústrias de corte de couro e metal-mecânica. A indústria confirma com os clientes que seus itens vão ser produzidos, esperando obter um determinado lucro com essa produção. Assim, a indústria faz o pedido do recipiente (peça de couro ou placa de metal), sabendo que ele poderá conter partes com defeitos (por exemplo, oriundos da extração, do manuseio ou do transporte), porém desconhece com precisão quais defeitos e suas localizações no recipiente. Quando o recipiente chega na indústria, ele passa por uma verificação de controle. Se o recipiente não apresenta defeitos, é certo que se produzirá todos os itens selecionados inicialmente. Por outro lado, na

[3] https://proceedings.science/sbpo-2020/papers/modelo-de-programacao-estocastica-para-um-problema-de-corte-de-itens-irregulares

[4] https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.13122

ocorrência dos defeitos, a indústria precisa reconfirmar quais itens virão a ser produzidos, de forma que alguns itens selecionados poderão ser cancelados e, neste caso, haverá um custo (por exemplo, multa) pela não produção desses itens (por exemplo, por não conseguir entregá-los dentro do prazo acordado com os clientes). Claramente que a indústria quer minimizar o impacto que os custos de cancelamento trazem e, por isso, ela desejaria tomar decisões mais acertadas sobre quais itens selecionar para poder confirmar com seus clientes e não ter que pagar multas.

Para o problema do Capítulo 4, propõe-se um modelo de programação estocástica de dois estágios com recurso. O primeiro estágio lida com as decisões sobre quais itens selecionar, enquanto o segundo estágio contém cenários com a realização dos defeitos no recipiente, sendo preciso decidir sobre o posicionamento dos itens no recipiente e sobre o possível cancelamento de itens. Os cenários modelam as possibilidades de ocorrência dos defeitos e suas localizações no recipiente, sendo associada uma probabilidade de ocorrência para cada cenário. Objetivando controlar a variabilidade que possa existir nas realizações dos defeitos entre os cenários, investiga-se também uma medida de risco visando obter soluções robustas para o problema. Experimentos computacionais são realizados sobre várias instâncias e diferentes casos de probabilidade para os cenários, buscando concluir sobre a necessidade de se considerar as incertezas no problema para obter soluções economicamente viáveis. Dessa forma, avaliam-se as soluções através do cálculo do valor esperado da informação perfeita e o valor da solução estocástica. As conclusões apontam que as incertezas sobre os defeitos impactam diretamente no lucro ao cortar a recipiente, sendo realmente vantajoso resolver um modelo de programação estocástica.

Alguns resultados iniciais do Capítulo 4, envolvendo o modelo de programação estocástica, foram publicados como artigo completo e apresentados de forma oral no *LIII Simpósio Brasileiro de Pesquisa Operacional - SBPO*, realizado entre os dias 03 e 05 de novembro de 2021 (SOUZA QUEIROZ; ANDRETTA, 2021). Esse artigo foi escrito em colaboração com Marina Andretta (ICMC-USP). Além disso, todo o capítulo foi escrito na forma de um artigo a ser submetido para um periódico internacional.

O Capítulo 5 contém as conclusões obtidas com o desenvolvimento da tese, além de sugestões de trabalhos que podem ser desenvolvidos no futuro como forma de estender os estudos aqui iniciados.

CAPÍTULO

2

HEURÍSTICAS PARA O PROBLEMA DA MOCHILA

Este capítulo traz heurísticas para o problema de corte bidimensional de itens irregulares. Utiliza-se o conceito de *inner-fit raster* e *no-fit raster* para o posicionamento viável de itens. A primeira heurística é um algoritmo genético de chaves aleatórias viciadas, ou seja, é baseada em população, e a outra é uma busca em vizinhança variável, ou seja, é de trajetória única. Nas heurísticas, uma solução é codificada através de um vetor e o posicionamento dos itens considera três regras inspiradas na *bottom-left*. Os experimentos numéricos realizados em instâncias da literatura mostram que as heurísticas propostas são melhores que o estado-da-arte, sendo possível obter soluções iguais ou melhores para todas as instâncias. Na média, a área ocupada aumentou em torno de 6,44% e foi possível obter a solução reconhecida ótima para 60% das instâncias. A heurística baseada em populações teve desempenho geral melhor, obtendo soluções com melhores taxas de ocupação em um tempo menor.

Resultados preliminares desse capítulo foram publicados na forma de um artigo completo e apresentados na *XLIV Conferência Latino-americana de Informática* (SOUZA QUEIROZ; MUNDIM; ANDRETTA, 2018). Todo o capítulo foi publicado no periódico *Applied Soft Computing* (SOUZA QUEIROZ; ANDRETTA, 2020b).

2.1 Introdução

Problemas de corte e empacotamento estão relacionados com o corte de recipientes ou o empacotamento de itens, para fins de otimizar algum objetivo. Os problemas de corte e empacotamento são de grande importância prática, pois são comumente encontrados em empresas e indústrias, como na metal-mecânica, na automotiva e de máquinas agrícolas, na têxtil, na de produção de móveis, no transporte logístico e fretamento (SCHEITHAUER, 2017). Além disso, diversos problemas de corte e empacotamento são NP-Difíceis (GAREY; JOHNSON,

1979), justificando a aplicação de heurísticas.

Neste trabalho, resolve-se o problema da mochila bidimensional com itens irregulares (*Two-dimensional Irregular Knapsack Problem - 2IKP* (WÄSCHER; HAUSSNER; SCHUMANN, 2007). Nesta versão do 2IKP, tem-se um conjunto de itens irregulares e um recipiente retangular de dimensões fixas. O objetivo é determinar um padrão de corte viável que tenha a maior área ocupada. Um padrão de corte viável é um posicionamento de um subconjunto de itens onde não há sobreposição e os itens estão inteiramente dentro do recipiente.

Algumas aplicações que o 2IKP pode ter são: indústria automotiva, na produção de assentos; agroindústria, no corte de placas de aço; indústria têxtil e de confecção de couro, na confecção de roupas, bolsas e calçados; indústria do esporte, na produção de bolas; setor de logística, no carregamento de veículos; indústria moveleira, no corte de placas de madeira (ALVES *et al.*, 2012b; BALDACCI *et al.*, 2014; PINTO *et al.*, 2016). Além disso, o problema da mochila com itens irregulares também aparece como um subproblema de outros, por exemplo, ao resolver os problemas de empacotamento em recipientes e de corte de estoque com métodos baseados em geração de colunas (VALLE *et al.*, 2012; SONG; BENNELL, 2014b).

Devido à geometria dos itens, a verificação de sobreposição entre itens tem sido objeto frequente de estudo na literatura de corte com itens irregulares (OLIVEIRA; FERREIRA, 1993; BURKE *et al.*, 2010; CHERNOV *et al.*, 2012). Segundo Bennell e Oliveira (2008), as ferramentas mais comuns para assegurar a não sobreposição são os métodos: *raster*, que discretiza os itens e recipiente em malhas; *phi-function*, que usa expressões matemáticas que fornecem a posição relativa entre itens; trigonometria direta, que testa a interseção entre segmentos de reta e inclusão de pontos; e, *no-fit polygon*, que é um polígono construído sobre cada par de itens indicando em seu interior o estado de sobreposição. Outras ferramentas surgiram da combinação desses métodos, como é o caso do *no-fit raster*, que usa os métodos *raster* e o *no-fit polygon* (TOLEDO *et al.*, 2013; MUNDIM; ANDRETTA; QUEIROZ, 2017).

Na literatura de problemas com itens irregulares, há poucos resultados envolvendo o 2IKP (MUNDIM *et al.*, 2018). A grande porcentagem dos trabalhos é sobre problemas com uma dimensão aberta. Neste problema é dado um conjunto de itens e um recipiente com uma dimensão variável, objetivando minimizar a parte usada da dimensão variável do recipiente dado o posicionamento de todos os itens (ALBANO; SAPUPPO, 1980; DOWSLAND; DOWSLAND; BENNELL, 1998; GOMES; OLIVEIRA, 2006; ELKERAN, 2013; CHERRI *et al.*, 2016; PERALTA; ANDRETTA; OLIVEIRA, 2018). Para problemas com duas dimensões abertas, com bem menos trabalhos publicados, pode-se mencionar os modelos de programação não linear de Kallrath (2009) e Bennell *et al.* (2015). Em Mundim, Andretta e Queiroz (2017) há um algoritmo genético de chaves aleatórias viciadas que usa a heurística *bottom-left* para construir padrões sobre as sequências informadas pelos cromossomos.

Com relação aos métodos de solução para o 2IKP e suas variantes, Scheithauer e Terno (1993) propuseram modelos de programação linear inteira mista, em que as restrições de não

sobreposição são derivadas a partir do *no-fit polygon*. Crispin *et al.* (2005) desenvolveram algoritmos genéticos para um problema na indústria de calçados. Os algoritmos usam o *no-fit polygon* para determinar posições que maximizam a utilização do recipiente ou que mantêm os itens em contato. Martins e Tsuzuki (2010) propuseram uma heurística baseada em recozimento simulado para os casos com recipientes retangulares e irregulares, assumindo que os itens irregulares possuem rotação livre. Os autores calculam uma região livre de colisão usando o *no-fit polygon* para determinar o local de posicionar os itens. A solução inicial da heurística é gerada de forma aleatória e novas soluções são obtidas movendo itens de suas posições.

Valle *et al.* (2012) apresentaram heurísticas baseadas em GRASP. A solução inicial é gerada de forma aleatória gulosa a partir de uma lista de itens candidatos. Soluções melhores são obtidas fazendo a permuta de itens da lista, de forma a considerar uma nova sequência para empacotar os itens. Para empacotar um item, verifica-se as posições viáveis a partir do *no-fit polygon* entre o item e os itens já empacotados. Alves *et al.* (2012b), no corte de peças de couro para a indústria automobilística, propuseram uma heurística de busca em vizinhança variável que considera a busca sobre a sequência de itens. Foram propostos quatro operadores para mover itens dentro da sequência em busca de uma sequência que resulte em uma solução melhor. Dalalah, Khrais e Bataineh (2014) consideraram recipientes retangulares e irregulares, propondo uma heurística construtiva, que empacota os itens conforme a utilização da envoltória convexa e faz a união de itens para melhor aproveitamento do recipiente.

Baldacci *et al.* (2014), com aplicação na indústria de couro, consideraram a representação do recipiente sobre uma malha, apresentado um modelo linear inteiro e uma relaxação lagrangeana. Os autores também apresentaram heurísticas a partir da relaxação proposta, que consideram formas distintas para empacotar os itens. Pinto *et al.* (2016) melhoraram os resultados de Alves *et al.* (2012b), a partir da proposta de uma heurística construtiva que simula o empacotamento de itens em pontos do recipiente, selecionando o item e o ponto que geram a melhor ocupação. Os autores ainda propuseram uma busca local, em que itens já empacotados são removidos de suas posições e itens ainda não empacotados são testados nessas posições. Mais recentemente, Mundim *et al.* (2018) propuseram uma heurística geral que escolhe aleatoriamente entre seis regras de posicionamento, inspiradas na *bottom-left*, para construir soluções sobre sequências aleatórias de itens. Cherri *et al.* (2019) propuseram modelos de programação por restrições, além de uma restrição global que garante a não sobreposição de itens.

As contribuições deste trabalho baseiam-se em dois métodos eficazes para o 2IKP, que são baseados no algoritmo genético de chaves aleatórias viciadas e na busca em vizinhança variável. Eles foram desenvolvidos observando a abordagem "menos é mais" (*less is more*), em que bons resultados podem ser obtidos sem a necessidade de componentes ou etapas sofisticadas (COSTA; ALOISE; MLADENOVIC, 2017). Segue-se o *framework* padrão já proposto para eles, em que o paralelismo não foi considerado ou explorado. Esses métodos são atraentes em comparação a outros, pois são simples de serem implementados por profissionais e pesquisadores. Eles

permitem diferentes maneiras de codificar e decodificar uma solução devido às suas estruturas internas e o fato de novos componentes serem facilmente integrados. Eles também foram aplicados com sucesso na resolução de vários problemas de otimização com um e múltiplos objetivos (GONÇALVES; RESENDE, 2011; MLADENOVIĆ; HANSEN, 1997).

Um levantamento sobre as aplicações do algoritmo genético de chaves aleatórias viciadas foi conduzido por Gonçalves e Resende (2011), enquanto um levantamento relacionado à busca em vizinhança variável pode ser encontrado em Hansen, Mladenović e Pérez (2010). Alguns problemas de otimização em que esses métodos foram recentemente aplicados para resolver são o roteamento de veículos (QIU *et al.*, 2018; RUIZ *et al.*, 2019), escalonamento (LI; PAN; WANG, 2014), localização de facilidades (BIAJOLI; CHAVES; LORENA, 2019), projeto de redes de telecomunicações (ANDRADE *et al.*, 2015), entre outros (HERRÁN; COLMENAR; DUARTE, 2019; PEKEL; KARA, 2019). Em relação à sua aplicação em problemas de corte e empacotamento, com itens de formato regular, tem-se o algoritmo genético de chaves aleatórias viciadas de GONÇALVES e Resende (2013), para os problemas de empacotamento bidimensional e tridimensional; Lalla-Ruiz e Voß (2015), para o problema de atribuição de múltiplas mochilas; e Hottung e Tierney (2016), para um problema de carregamento de contêineres. No caso de itens irregulares, pode-se citar os algoritmos genéticos de Silveira (2013), Pinheiro, Amaro Jr. e Saraiva (2016), Mundim, Andretta e Queiroz (2017) e Amaro Jr., Pinheiro e Coelho (2017). No que diz respeito à busca em vizinhança variável, tem-se Hifi e Wu (2015), que resolveu um problema da mochila multidimensional; Zeng *et al.* (2018), com o empacotamento de círculos de diferentes raios em quadrados; e Santos *et al.* (2019), que lidou com uma nova variante do problema de empacotamento em recipientes, com categorias. No caso de itens irregulares, pode-se citar as versões de Alves *et al.* (2012a) e Alves *et al.* (2012b).

O algoritmo genético de chaves aleatórias viciadas considera a otimização sobre múltiplas trajetórias (GONÇALVES; RESENDE, 2011). Implementa-se esse algoritmo de forma que os indivíduos da população contenham informações sobre a ordem e a rotação para empacotar os itens, bem como da probabilidade usada na fase de cruzamento, fazendo com que esse parâmetro seja variável no decorrer das gerações. A busca em vizinhança variável considera a otimização sobre uma trajetória única (MLADENOVIĆ; HANSEN, 1997). Nessa heurística ocorre a mudança sistemática de vizinhanças a medida que a solução fica estagnada em um ótimo local. Propõem-se estruturas de vizinhança baseadas em movimentos de troca e inserção.

Para ambas as heurísticas, adota-se a representação da solução do problema sobre um vetor de itens. Com esse vetor, constrói-se a solução aplicando regras de posicionamento, sendo uma delas a *bottom-left*. A aplicação das regras é baseada na frequência daquela que tem gerado as melhores soluções. Os itens são empacotados em uma malha que representa o recipiente, usando o conceito de *no-fit raster*. Algumas posições da malha podem ser "puladas" como forma de diversificar a busca e escapar de ótimos locais durante o posicionamento dos itens. A validação das heurísticas é feita sobre instâncias da literatura, mostrando que ambas são competitivas entre

si e capazes de melhorar as soluções de outros métodos recentes da literatura.

Este trabalho está estruturado da seguinte forma: a Seção 2.2 descreve o problema e a forma como se verifica a viabilidade de uma solução; a Seção 2.3 descreve as duas heurísticas propostas e suas estruturas, incluindo a forma como uma solução é representada e as regras de posicionamento; a Seção 2.4 apresenta os experimentos computacionais sobre um conjunto de instâncias da literatura; e a Seção 2.5 apresenta as principais conclusões do trabalho.

2.2 Definição do problema

No 2IKP, o recipiente é representado no primeiro quadrante do Plano Cartesiano, com origem no ponto $(0,0)$. O eixo x está associado à largura e o eixo y ao comprimento. No problema é dado um conjunto de itens I, sendo que cada item $i \in I$ é caracterizado por um polígono convexo ou não-convexo. Cada item i é descrito por um conjunto de vértices ordenados (no sentido anti-horário), sendo um destes vértices tomado como o vértice de referência, que é por onde o item é posicionado no recipiente. Cada item i tem uma área a_i, um valor v_i, que neste trabalho é igual a área, um conjunto de possíveis rotações θ_i. Também é dado um recipiente B, que tem o formato retangular, com largura L e comprimento W. O objetivo é obter um padrão de corte viável, que tenha o valor máximo, de um subconjunto de itens de I. Um padrão viável implica que os itens estão posicionados de forma a não ter sobreposição e inteiramente dentro do recipiente. Nota-se que um padrão viável que contém todos os itens de I é sempre ótimo para o 2IKP.

Devido à geometria dos itens, a verificação da sobreposição entre itens é feita empregando o *no-fit raster*. Inicialmente, calcula-se o *no-fit polygon* entre todos os pares de itens i e j de I, em cada combinação de suas rotações, utilizando o algoritmo de CUNINGHAME-GREEN (1989). Caso algum item seja não-convexo, faz-se a sua decomposição em polígonos convexos (por meio de triangulação) e o algoritmo é aplicado sobre cada polígono convexo para, então, realizar a união dos *no-fit polygons* parciais e obter o *no-fit polygon* final entre os itens i e j para a rotação $r \in \theta_i$ e $s \in \theta_j$, representado por $NFP_{ij,rs}$. Para obter o *no-fit raster*, discretiza-se o $NFP_{ij,rs}$ sobre uma matriz cujo interior é preenchido com o valor "um" para indicar que o item j não pode ser posicionado ali. Na discretização do $NFP_{ij,rs}$, adota-se uma distância inteira positiva g entre duas coordenadas consecutivas na direção da largura e do comprimento.

Para assegurar que os itens não extrapolem as dimensões do recipiente, utiliza-se o conceito de *inner-fit raster*. Inicialmente, calcula-se o *inner-fit polygon* para cada item i, em cada uma de suas rotações $r \in \theta_i$, sobre o recipiente B. Esse polígono é obtido ao transladar i na rotação r, pelo seu vértice de referência, de maneira que i sempre esteja encostando nas bordas de B, resultando no $IFP_{i,r}$. Em seguida, obtém-se a matriz para o $IFP_{i,r}$ a partir da sua discretização conforme a distância g. As posições da matriz em que o item i não pode ser empacotado são preenchidas com o valor "um".

2.3 Métodos para o 2IKP

Os métodos propostos para o 2IKP são discutidos a seguir. O primeiro é baseado no algoritmo genético de chaves aleatórias viciadas (*Biased Random-Key Genetic Algorithm - BRKGA*) *BRKGA*Biased Random-Key Genetic Algorithm* e o outro na busca em vizinhança variável (*Variable Neighborhood Search - VNS*) *VNS*Variable Neighborhood Search*. Ambos os métodos representam a solução do problema por um vetor e usam as mesmas regras de posicionamento. As regras de posicionamento geram uma lista ordenada de pontos onde os itens podem ser posicionados no recipiente. Dependendo de como os pontos desta lista são ordenados, o padrão de corte resultante pode diferir significativamente de outros, o que ajuda a explorar e alcançar soluções de qualidade. Além disso, propõe-se pular pontos viáveis ao posicionar itens para diversificar a busca e escapar de soluções ótimas locais.

O 2IKP é um problema difícil, porém com relativamente poucas contribuições na área, motivando o desenvolvimento de novos métodos de solução. Como o BRKGA e o VNS têm se mostrado competitivos na resolução de diferentes problemas de otimização combinatória, inclusive problemas de *nesting*, eles são aplicados para o 2IKP e comparados com outros métodos eficazes da literatura. Além disso, por se tratarem de paradigmas diferentes, um baseado em população e outro em trajetória única, a comparação entre eles também segue como umas das contribuições, em especial, para contextos práticos do problema.

2.3.1 BRKGA

No BRKGA, cada cromossomo é representado por um vetor c de tamanho n, cujas componentes são chaves aleatórias (isto é, um número real entre 0 e 1). A população inicial consiste de P cromossomos gerados aleatoriamente. Em seguida, cada cromossomo $c \in P$ passa por um decodificador, em que o vetor de chaves aleatórias é transformado em uma solução do problema e, assim, obtém-se a aptidão $f(c)$. Em seguida, a população é dividida em dois grupos, um elite P_e e outro não-elite $P_{\bar{e}}$, tal que $P = P_e \cup P_{\bar{e}}$. O grupo elite contém os indivíduos de maior aptidão (GONÇALVES; RESENDE, 2011).

A população P da próxima geração mantém o grupo P_e sem modificações. O restante dos indivíduos de P advém de dois outros grupos, P_m e P_f. O grupo P_m representa os indivíduos mutantes, que são cromossomos gerados aleatoriamente. O grupo P_f consiste de indivíduos descendentes (filhos). Um indivíduo descendente $c_d \in P_f$ é gerado a partir de um indivíduo $c_e \in P_e$ e outro $c_{\bar{e}} \in P_{\bar{e}}$, considerando a probabilidade viciada ρ_e de herdar chaves de c_e. A i-ésima componente $c_d[i]$ vem de $c_e[i]$ com probabilidade ρ_e ou de $c_{\bar{e}}[i]$ com probabilidade $1-\rho_e$. A população P da próxima geração é a união de P_e, P_m e P_f, sendo sempre de tamanho constante durante o processo de evolução.

Os valores utilizados pelo BRKGA para os seus parâmetros são constantes e fornecidos como entrada. São eles: quantidade de chaves n, quantidade máxima de gerações max, tamanho

da população P, tamanho da população elite P_e, tamanho da população mutante P_m, probabilidade de herdar chaves de pais elites ρ_e. Observa-se que o BRKGA necessita de um bom decodificador e da correta calibração de seus parâmetros para que boas soluções sejam obtidas ao final do processo evolutivo. A estrutura geral de um BRKGA é dada no Algoritmo 2.1.

Algoritmo 2.1: Estrutura geral do BRKGA.

```
1 início
2     Gerar aleatoriamente uma população inicial P.
3     enquanto não atingir o critério de parada faça
4         Decodificar cada cromossomo de P e obter sua aptidão.
5         Ordene os cromossomos de P em ordem não crescente de aptidão e coloque em
            P_e os melhores cromossomos.
6         Gere aleatoriamente P_m novos cromossomos.
7         Gerar P_f cromossomos descendentes utilizando um cruzamento uniforme
            parametrizado, para um pai aleatório de P_e e outro de P_ē = P \ P_e.
8         P ← P_e ∪ P_m ∪ P_f.
9     retorna cromossomo de P com a melhor aptidão.
```

O BRKGA para o 2IKP parte do Algoritmo 2.1, considerando a interface computacional de Toso e Resende (2015). Propõe-se que a probabilidade ρ_e não seja um valor constante, mas sim uma característica do indivíduo, isto é, ela é uma chave aleatória no cromossomo. Cada cromossomo c da população é um vetor de n chaves, em que $n = |I| + N + 1$. Isso significa que n é igual ao número de itens em I, o número N de posições viáveis que podem ser "puladas" ao posicionar itens (ou seja, pontos da malha que são viáveis para posicionar itens, mas podem ser ignorados) e a probabilidade viciada ρ_e, que é usada na etapa de cruzamento caso c seja um indivíduo elite.

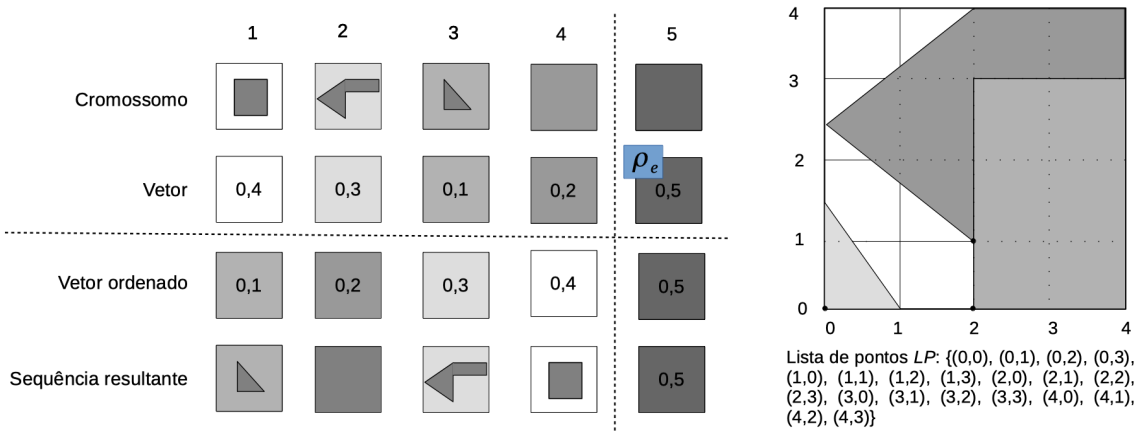
Em cada cromossomo c, as componentes de 1 até $n-1$ estão indexadas com os itens de I e a quantidade N de posições viáveis a pular. Além disso, ao invés de adotar uma probabilidade viciada ρ_e constante, utiliza-se o valor da chave aleatória na última (n-ésima) componente para ser a probabilidade do descendente herdar chaves do cromossomo c, caso c faça parte do grupo elite. Ao tornar a probabilidade ρ_e dinâmica e dependente do cromossomo, bem como usando o número N de pontos viáveis para pular, pretende-se diversificar a exploração do espaço de busca e escapar de ótimos locais. Experimentos computacionais preliminares com essas estratégias indicaram a possibilidade de obter rapidamente bons padrões de corte. Além disso, ambas são contribuições desse trabalho em comparação com as outras heurísticas da literatura para problemas de *nesting*.

É importante mencionar que a proposta de pular pontos quando posicionando um item se deve à geometria irregular dos itens. Observou-se a partir de experimentos preliminares que itens podem ser combinados com precisão se alguns itens não forem posicionados no primeiro ponto

viável da lista ordenada (por exemplo, como feito nos trabalhos anteriores da literatura), mas em vez disso, em outros pontos viáveis. Em cada cromossomo, as chaves aleatórias associadas ao número N representam cada uma um ponto viável a ser ignorado. Portanto, se essas chaves aleatórias aparecerem apenas no final de um cromossomo, nenhum ponto viável será ignorado durante o posicionamento. Caso contrário, o número dessas chaves aleatórias que aparecem antes de um item indica quantos pontos possíveis são ignorados ao posicionar tal item (o exemplo na Figura 2.1 tem mais detalhes). Percebe-se que esta característica não é bem explorada na literatura de problemas de corte e empacotamento com itens de formato irregular. Na literatura relacionada é comum usar apenas uma regra de posicionamento, especialmente a *bottom-left*, e os itens são sempre empacotados no primeiro ponto viável da lista ordenada. Como uma tentativa de explorar diferentes padrões de corte, pode-se usar muitas regras de posicionamento; no entanto, isso pode aumentar o tempo computacional e resultar em padrões de corte não tão satisfatórios devido à dificuldade em escolher uma regra de posicionamento adequada.

O decodificador para o 2IKP está descrito no Algoritmo 2.2. Denomina-se o BRKGA desenvolvido para o 2IKP como BRKGA$_{2IKP}$. A Figura 2.1 exemplifica um cromossomo e o padrão de corte (isto é, solução) obtido. Este exemplo considera três itens, sem rotação, e o número N é igual a 1, indicando que no máximo um ponto viável pode ser ignorado ao posicionar algum item, resultando em um cromossomo com $n = 5$ componentes. Após ordenar as chaves aleatórias em ordem não decrescente (observe que a chave do último componente, que contém o valor de ρ_e, não é considerada na ordenação), a sequência para o posicionamento dos itens é obtida. Na sequência resultante, o item 3 é posicionado no primeiro ponto viável de LP, que é o ponto $(0,0)$. Como resultado, os pontos $(0,0)$ e $(0,1)$ tornam-se inviáveis para posicionar outros itens, por isso são retirados da lista ordenada LP. A seguir, entre os itens 3 e 2, existe uma chave aleatória associada ao número N. Isso significa que o item 2 não é posicionado no primeiro ponto viável de LP, que seria o ponto $(2,0)$, mas deve ser posicionado no segundo ponto viável, que é o ponto $(2,1)$. Observe que o número de chaves aleatórias associadas a N antes de um item representa o número de pontos possíveis a serem ignorados ao posicionar tal item. Como resultado do posicionamento do item 2, os seguintes pontos são removidos da lista (uma vez que se tornam inviáveis para outros itens): $(1,2),(1,3),(2,1),(2,2),(2,3)$ e $(3,3)$. Em seguida, o item 1 é posicionado no primeiro ponto viável de LP, que é o ponto $(2,0)$.

O Algoritmo 2.2 inicia copiando as $n-1$ primeiras chaves aleatórias do cromossomo c para um vetor Or. Essas chaves contemplam os itens de I e a quantidade N de posições viáveis a pular. Em seguida, o vetor Or é ordenado de forma não decrescente pelo valor das chaves aleatórias. Ao realizar essa ordenação, a sequência inicial dos itens é modificada para uma nova sequência, embaralhando itens de I com posições a pular de N (veja na Figura 2.1(a)). Como várias sequências são geradas no decorrer do algoritmo, utiliza-se uma tabela *hash* para armazenar as sequências Or já analisadas e evitar re-cálculos. A chave para a busca na tabela *hash* consiste em um vetor de caracteres com a sequência resultante de Or. Na linha 5, caso Or já tenha sido analisado, recupera-se a solução e sua aptidão da tabela *hash*, e o algoritmo finaliza.



(a) Decodificação de um cromossomo (b) Padrão de corte resultante

Figura 2.1 – Representação de um cromossomo do BRKGA$_{2IKP}$.

Os itens no Algoritmo 2.2 são posicionados conforme a sequência em Or e a ordem dos pontos em LP (laço das linhas 6-28). A lista ordenada de pontos LP é criada por uma das regras de posicionamento na Seção 2.3.3. Aplica-se primeiro a regra que tenha gerado até o presente o maior número de padrões de corte melhores, conforme armazenado em $Rank$, e assim por diante. Observe pela condição da linha 25, que no laço associado a segunda ($r = 2$) e terceira ($r = 3$) regras, caso seja detectado que padrão de corte Sol_r será pior (isto é, de valor menor) do que o padrão Sol_1 (da primeira regra), reinicia-se o laço com a próxima regra, assim evitando computação desnecessária. Dentre as regras aplicadas por completo, aquela que gerou o melhor padrão de corte é identificada na variável B_r (linha 28) e usada para atualizar o $Rank$ (linha 29). Além disso, a melhor solução encontrada (linha 28) é armazenada na tabela *hash* (linha 30). A função $f()$ retorna o valor total dos itens no dado padrão de corte.

No laço das linhas 9-26 do Algoritmo 2.2, realiza-se o posicionamento dos itens no recipiente seguindo o ordem dos pontos na lista LP. Inicia-se o laço verificando se o elemento i é ou não um item de I (linha 11). Caso não seja, incrementa-se o valor da variável S, que armazena quantos pontos viáveis de LP deverão ser pulados quando for posicionar um próximo item de I (linhas 12-13). Caso i seja um item de I, identifica-se a primeira rotação θ que se deve testar para o posicionamento de i. Para tanto, usa-se o valor chave aleatória associada a i, que está na posição $Or[j]$ (linha 14). O número de rotações para i, que é $|\theta_i|$, indica em quantos subintervalos consecutivos o intervalo $[0,1)$ vai ser particionado. Por exemplo, caso o item i possa ser empacotado nas rotações 0 e 90 graus, então $|\theta_i| = 2$ e, assim, há dois intervalos: $[0,1/2)$, que está associado à rotação de 0 grau, e $[1/2,2/2)$, que está associado à rotação de 90 graus.

Na Linha 16 do Algoritmo 2.2, testa-se a viabilidade de posicionar o item i, dada a rotação θ, no ponto (a,b) de LP. Os pontos de LP são percorridos na ordem em que eles foram gerados pela regra de posicionamento, tal que se busca o primeiro ponto viável. Caso não haja

Algoritmo 2.2: Decodificador do BRKGA$_{2IKP}$.

Entrada: cromossomo c de tamanho $n = |I| + N + 1$; itens I; vetor *Rank* de *ranking* das regras de posicionamento; tabela *hash* T.
Saída: B_{sol}, que é um padrão de corte viável.

1 **início**
2 $B_{sol} \leftarrow \emptyset$; $B_r \leftarrow 0$; $Or[\ldots] \leftarrow c[1,\ldots,n-1]$.
3 $Or \leftarrow$ ordenar Or de forma não decrescente.
4 **se** *Or está na tabela* hash T **então**
5 **retorna** *A solução de Or armazenada em* T.
6 **para** $r \leftarrow 1,2,3$ **faça**
7 $LP \leftarrow$ aplicar a regra de posicionamento conforme $Rank[r]$ e obter uma lista ordenada de posições.
8 $j \leftarrow 1$; $S \leftarrow 0$; $Sol_r \leftarrow \emptyset$.
9 **enquanto** $j \leq n-1$ **faça**
10 $i \leftarrow Or[j]$.
11 **se** *i é uma posição vazia (isto é, posição a pular relacionada a N)* **então**
12 $S \leftarrow S+1$; $j \leftarrow j+1$.
13 Ir para a Linha 9.
14 $\theta \leftarrow$ a rotação de i dado o intervalo em que o valor da chave $Or[j]$ está.
15 **para** *cada ponto* (a,b) *de LP na ordem dada* **faça**
16 **se** *posicionamento é viável para i em* (a,b) *na rotação* θ **então**
17 **se** $S \neq 0$ **então**
18 $S \leftarrow S-1$.
19 Ir para a Linha 15.
20 $Sol_r \leftarrow Sol_r \cup \{i, \theta, (a,b)\}$.
21 Remover de LP os pontos sobrepostos ao empacotar i em (a,b) na rotação θ.
22 $S \leftarrow 0$; $j \leftarrow j+1$.
23 Ir para a Linha 9.
24 Atualizar θ para uma rotação em θ_i que ainda não foi testada e Ir para a Linha 16.
25 **se** $r > 1$ ***AND*** Sol_r *não poderá ser melhor do que* Sol_1 **então**
26 Ir para a Linha 6.
27 **se** $f(Sol_r) > f(B_{sol})$ **então**
28 $B_{sol} \leftarrow Sol_r$; $B_r \leftarrow r$.
29 Incrementar $Rank[B_r]$ em uma unidade e ordenar *Rank* de forma não crescente.
30 Armazenar a solução B_{sol} na tabela *hash* T.
31 **retorna** B_{sol}.

qualquer ponto viável para i na rotação θ, tenta-se novamente para uma outra rotação em R_i (linha 24), até testar todas as rotações em R_i. Caso não seja possível posicionar i em qualquer uma de suas rotações, o item i é deixado de fora, sendo possível identificar se o padrão Sol_r será pior do que Sol_1 (linha 25) por meio da soma do valor dos itens em Sol_r com o valor dos itens que ainda serão analisados em Or. Quando é possível empacotar i em alguma rotação θ, decrementa-se o valor de S, até que S seja 0 (linhas 18-19). Só assim, i poderá ser posicionado no padrão Sol_r (Linha 20). Na linha 21, os pontos da lista LP cobertos após o posicionamento de i são removidos e o algoritmo volta para o início do laço na linha 9 considerando o próximo elemento de Or.

2.3.2 VNS

A VNS é uma meta-heurística de trajetória única, que foi proposta por Mladenović e Hansen (1997). A VNS explora o espaço de soluções através de trocas sistemáticas de estruturas de vizinhança N_k, para um total de K_{max} vizinhanças. A VNS parte do pressuposto que um ótimo local em uma vizinhança não é necessariamente um ótimo local para outra(s) e um ótimo global é um ótimo local em relação a todas as vizinhanças.

A estrutura geral da VNS é descrita no Algoritmo 2.3. Inicia-se gerando aleatoriamente uma solução s. No laço das linhas 5-12, partindo da primeira estrutura de vizinhança, para $k = 1$, obtém-se aleatoriamente uma outra solução, s', considerando a vizinhança N_k sobre s (linha 6). Em seguida, essa solução s' passa por uma busca local (linha 7). Neste trabalho, considera-se a busca local realizada pela busca em descida com vizinhança variável (*Variable Neighborhood Descent - VND*) , que está no Algoritmo 2.4, resultando na versão Geral da VNS (GVNS) . Se a solução s'', resultante da busca local, for melhor do que s, então se atualiza s e reinicia-se com a busca sobre a primeira estrutura de vizinhança. Caso contrário, parte-se para a próxima estrutura de vizinhança (linha 12). O término do algoritmo ocorre ao atingir o critério de parada, de forma que a solução final é ótima local com relação a todas as estruturas de vizinhança.

Na VND descrita no Algoritmo 2.4, a solução s fornecida como entrada é explorada inicialmente sobre a primeira estrutura de vizinhança (linha 2). Neste caso, a nova solução s' consiste na melhor solução que pode ser obtida na vizinhança N_k sobre s (linha 5). Se a nova solução é melhor do que a corrente, atualiza-se a solução corrente s e reinicia-se a busca sobre a primeira estrutura de vizinhança (linha 8). Caso contrário, passa-se para a próxima estrutura de vizinhança (linha 10). A ordem em que as estruturas de vizinhança são visitadas impacta na convergência do algoritmo, sendo preferível que as estruturas iniciais sejam definidas sobre operações simples.

Na GVNS para o 2IKP, isto é, GVNS_{2IKP}, uma solução s é representada por um vetor de inteiros com n componentes, em que $n = |I| + N$, isto é, contempla o total de itens e a quantidade N de posições viáveis a pular. Similar ao BRKGA_{2IKP}, o vetor tem I componentes contendo números inteiros, cada um representando um item de I. As demais N componentes

Algoritmo 2.3: Estrutura geral da VNS.

```
1  início
2      Gerar uma solução inicial s de forma aleatória.
3      enquanto não atingir o critério de parada faça
4          k ← 1.
5          enquanto k ≤ K_max faça
6              s' ← obter uma solução aleatória na vizinhança N_k(s).
7              s'' ← Busca-Local(s', K_max).
8              se f(s'') > f(s) então
9                  s ← s''.
10                 k ← 1.
11             senão
12                 k ← k+1.
13     retorna solução s.
```

Algoritmo 2.4: Busca-Local definida sobre a VND.

```
1  início
2      s ← solução passada como entrada.
3      k ← 1.
4      enquanto k ≤ K_max faça
5          s' ← obter a melhor solução na vizinhança N_k(s).
6          se f(s') > f(s) então
7              s ← s'.
8              k ← 1.
9          senão
10             k ← k+1.
11     retorna solução s.
```

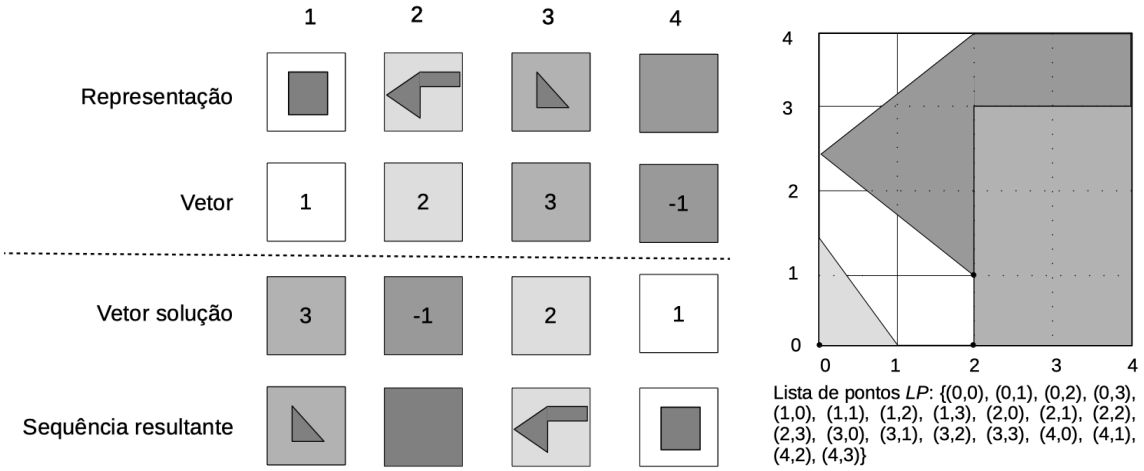
estão relacionadas com a quantidade de posições viáveis a pular, cada uma contendo o valor -1. Note que na representação da GVNS$_{2IKP}$ não existe uma (última) posição para armazenar uma probabilidade viciada.

A avaliação de uma solução s na GVNS$_{2IKP}$ é feita pela função de aptidão $f(s)$. Neste caso, transforma-se a solução s, representada por um vetor de inteiros, em um padrão de corte viável. Para este fim, utiliza-se o decodificador do BRKGA$_{2IKP}$ descrito no Algoritmo 2.2, com as seguintes modificações:

- na linha 2, considera-se $Or[\ldots] \leftarrow s[1,\ldots,n]$ no lugar de $Or[\ldots] \leftarrow c[1,\ldots,n-1]$;
- remove-se a linha 3;
- na linha 9, considera-se n ao invés de $n-1$;

- substitui-se a linha 14 por $rot \leftarrow 1 + mod(j, |\theta_i|)$ e $\theta \leftarrow \theta_i[rot]$. Ou seja, a primeira rotação na qual o item i vai ser testado advém da posição j em que i se encontra no vetor solução.

A Figura 2.2 mostra um exemplo de um vetor correspondente a uma solução e seu padrão de corte para a GVNS$_{2IKP}$. Considera-se o mesmo exemplo dado na Figura 2.1, com três itens, sem rotação, e o número N igual a 1, resultando em um vetor com $n = 4$ componentes. A sequência de posicionamento dos itens é obtida iterando-se sobre o vetor. Portanto, o item 3 é posicionado no primeiro ponto viável de LP, que é $(0,0)$. Entre os itens 3 e 2 existe um componente com o valor -1; ou seja, um ponto viável de LP deve ser ignorado. Então, o item 3 não é posicionado no primeiro ponto viável de LP, que é $(2,0)$, mas deve ser posicionado no segundo ponto viável, que é $(2,1)$. Finalmente, o item 1 é posicionado no primeiro ponto viável de LP, que é $(2,0)$.



(a) Vetor de uma solução (b) Padrão de corte resultante

Figura 2.2 – Representação de uma solução na GVNS$_{2IKP}$.

Dada a representação assumida para uma solução s na GVNS$_{2IKP}$, as estruturas de vizinhança são definidas sobre operações, em particular, de inserção e de troca de elementos de suas respectivas posições no vetor solução. Duas estruturas de vizinhança são consideradas, sendo que N_1 é definida pela operação de Troca, isto é, trocar dois elementos de posição, com as posições escolhidas aleatoriamente. A estrutura N_2 é definida sobre a operação de Inserção, que consiste em inserir o elemento da posição i imediatamente antes do elemento da posição j, em que as posições são escolhidas aleatoriamente.

A busca local baseada na VND usa as mesmas estruturas de vizinhança, N_1 e N_2. Isso significa que as operações de troca e inserção são testadas para todos as componentes do vetor solução, procurando a melhor solução na vizinhança atual. Portanto, na linha 5 do Algoritmo 2.4, dada a estrutura de vizinhança N_k, a melhor solução é obtida a partir de:

Busca local em N_1: operação de Troca em que todos as componentes i e j do vetor solução

são testadas. As duas componentes nas quais a troca resulta na melhor solução são então consideradas;

Busca local em N_2: operação de Inserção na qual todas as componentes i e j do vetor solução são testadas. Ao final, são consideradas as duas componentes em que a inserção resulta na melhor solução.

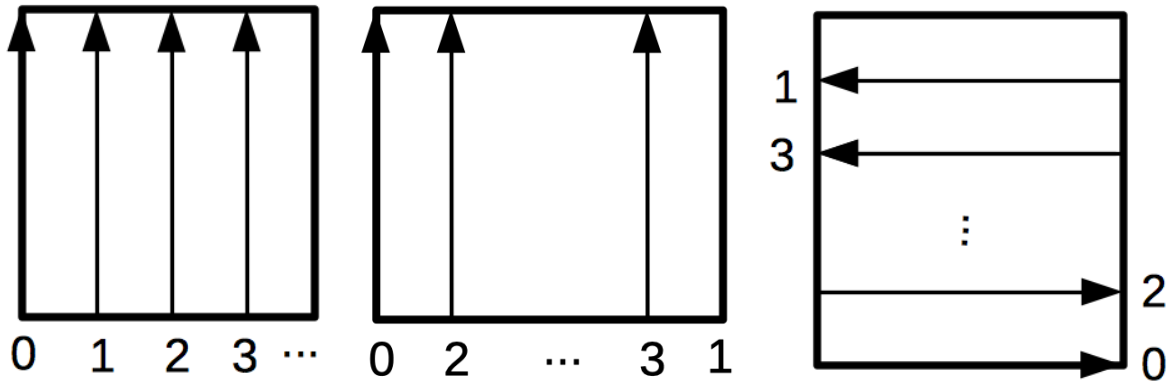
É importante destacar que outras estruturas testadas foram as operações de troca e inserção de sequências, ao invés de considerar apenas um elemento. As variantes que consideram as sequências na ordem inversa também foram testadas. Notou-se que essas estruturas requeriam demasiado tempo computacional (principalmente na fase de busca local), sem trazer melhoras significativas na solução final (na verdade, geralmente a solução final ficava pior do que quando considerando apenas as estruturas N_1 e N_2, devido ao tempo limite imposto como critério de parada), além de serem influenciadas diretamente pelo tamanho das sequências escolhidas. Desta forma, elas não foram consideradas na versão final da GVNS_{2IKP}.

2.3.3 Regras de posicionamento

Na literatura de problemas de corte e empacotamento, a regra *bottom-left* tem sido comumente adotada para o posicionamento de itens por geralmente trazer resultados satisfatórios. Na regra *bottom-left*, os itens devem ser posicionados mais à esquerda e abaixo possível. A aplicação de apenas uma regra de posicionamento pode restringir significativamente os padrões de corte gerados, como se observa nos resultados de Mundim, Andretta e Queiroz (2017). Por outro lado, a consideração de várias regras pode acarretar em um consumo desnecessário de tempo computacional e dificuldade em estabelecer um critério de escolha de qual regra aplicar, como se observa em Mundim *et al.* (2018).

Além de usar a regra *bottom-left*, descrita no Algoritmo 2.5, propõe-se duas outras regras de posicionamento, a partir dos resultados de Mundim *et al.* (2018). Na regra *bottom-left alternada*, a lista de pontos segue uma alternância entre mais a esquerda abaixo e mais a direta abaixo para posicionar os itens, indo pela esquerda, se b é par; caso contrário, indo pela direita, conforme descrito no Algoritmo 2.6. A outra regra é a *horizontal zig-zag alternada*, em que a lista de pontos segue um zig-zag horizontal para posicionar os itens, mais abaixo e indo da esquerda para a direita se o valor de a é par; caso contrário, mais acima e indo da direita para a esquerda, conforme descrito no Algoritmo 2.7. A Figura 2.3 ilustra a forma como os pontos dentro do recipiente são percorridos em cada regra de posicionamento.

Voltando ao exemplo das Figuras 2.1 e 2.2, se considerando que a primeira regra de posicionamento a ser testada é a *bottom-left*, obtém-se o padrão de corte em 2.1(b) e 2.2(b). Seguindo a sequência em 2.1(a) e 2.2(a), o primeiro item a ser testado é o 3, que deve ser posicionado na primeira posição viável, neste caso, ponto $(0,0)$. O segundo item é o 2 cuja primeira posição viável é o ponto $(2,0)$, porém, ele deve ser posicionado somente na segunda



(a) *Bottom-left* (b) *Bottom-left alternado* (c) *Horizontal zig-zag alternado*

Figura 2.3 – Regras de posicionamento usadas pelas heurísticas.

Algoritmo 2.5: *Bottom-left*.

Entrada: Recipiente $B = (L, W)$; distância g.
Saída: Lista ordenada de pontos LP.

```
1 início
2     LP ← ∅.
3     a ← 0; b ← 0;
4     enquanto a ≤ L faça
5         enquanto b < W faça
6             LP ← LP ∪ (a, b).
7             b ← b + g.
8         a ← a + g.
9     retorna LP.
```

Algoritmo 2.6: *Bottom-Left* alternada.

Entrada: Recipiente $B = (L, W)$; distância g.
Saída: Lista ordenada de pontos LP.

```
1 início
2     LP ← ∅.
3     c ← 0; f ← L − g; aux ← 0.
4     enquanto aux ≤ L faça
5         se aux é par então
6             a ← c; c ← c + g.
7         senão
8             a ← f; f ← f − g.
9         para b ← 0, g, 2g, 3g, ..., W − 1 faça
10            LP ← LP ∪ (a, b).
11        aux ← aux + g.
12    retorna LP.
```

Algoritmo 2.7: Horizontal zig-zag alternada.

Entrada: Recipiente $B = (L, W)$; distância g.
Saída: Lista ordenada de pontos LP.

início
  $LP \leftarrow \emptyset$.
  $c \leftarrow 0$; $f \leftarrow W - g$; $aux \leftarrow 0$.
  enquanto $aux < W$ **faça**
    se *aux é ímpar* **então**
      $b \leftarrow c$; $c \leftarrow c + g$.
      para $a \leftarrow 0, g, 2g, 3g, \ldots, L$ **faça**
        $LP \leftarrow LP \cup (a, b)$.
    senão
      $b \leftarrow f$; $f \leftarrow f - 1$.
      para $a \leftarrow L, \ldots, 3g, 2g, g, 0$ **faça**
        $LP \leftarrow LP \cup (a, b)$.
  retorna LP

posição viável, que é o ponto $(2,1)$. Por fim, tem-se o terceiro item, que é o item 1, sendo posicionado na primeira posição viável, neste caso, ponto $(2,0)$. Observe que se não fosse adotada a estratégia da quantidade N de posições viáveis a pular, não haveria uma permutação entre os três itens, independente de qualquer uma das três regras de posicionamento utilizadas, capaz de resultar no padrão em 2.1(b) e 2.2(b).

2.4 Experimentos computacionais

As heurísticas, BRKGA$_{2IKP}$ e GVNS$_{2IKP}$, foram codificadas na linguagem C++ e os experimentos ocorreram em um computador com processador Intel Xeon X5660 de 2,8 GHz, 48 GB de RAM e sistema Linux Ubuntu 16.04 LTS. Devido a aleatoriedade presente nas heurísticas, cada instância foi resolvida 10 vezes, com valores diferentes de semente.

2.4.1 Parâmetros e instâncias

Os parâmetros dos métodos foram calibrados com o pacote *irace* (*Iterated Race for Automatic Algorithm Configuration*) de Lopez-Ibanez *et al.* (2016). Este pacote é implementado no *software* estatístico R e usa uma corrida iterada para estimar os melhores parâmetros. Para tanto, fornecem-se intervalos para cada parâmetro, um limite de ajuste de 300 experimentos, um conjunto de instâncias de treinamento (incluindo as de pequeno, médio e grande porte) e os parâmetros a serem calibrados. Os intervalos de cada parâmetro foram escolhidos com base em estudos anteriores e alguns experimentos preliminares.

Para o BRKGA$_{2IKP}$, os parâmetros foram obtidos dos seguintes intervalos: tamanho

da população, $|P| \in [100,\ 500]$; tamanho da população elite $|P_e| \in [0,1,\ 0,3]|P|$; e tamanho da população mutante, $|P_m| \in [0,05,\ 0,25]|P|$. Portanto, de acordo com o *irace*, os seguintes valores podem produzir bons resultados: $|P| = 160$, $|P_e| = 0,25|P|$ e $|P_m| = 0,1|P|$. A GVNS$_{2IKP}$ possui como parâmetro apenas o critério de parada, visto que o total de estruturas de vizinhança K_{max} é igual a 2.

Com relação ao parâmetro N, que representa o número de pontos viáveis a serem ignorados ao posicionar um item, testou-se N no intervalo $[0,1,\ 1,0]|I|$ e obteve-se bons resultados para $N = \lfloor 0,25|I| \rfloor$. Percebeu-se que pequenos valores de N levam à baixa diversidade de soluções, uma vez que a maioria dos itens estavam dispostos nos primeiros pontos viáveis. Por outro lado, grandes valores de N levam a muitas soluções com áreas vazias nas quais itens (pequenos) poderiam ser posicionados, resultando em mais itens a serem deixados para trás.

Os métodos propostos possuem dois parâmetros como critérios de parada (o primeiro a ser alcançado os interrompe). Esses critérios foram escolhidos de acordo com a literatura, visando uma comparação justa com os métodos propostos em termos de solução e tempo de execução. Portanto, cada método para após atingir um limite de tempo de 300 segundos, que foi definido por Mundim *et al.* (2018), ou antes desse tempo, apenas se um padrão de corte ótimo conhecido for alcançado. Este padrão de corte surge quando todos os itens estão posicionados no recipiente ou a área do recipiente é 100% ocupada, já que se assume o valor igual a área do item.

As instâncias utilizadas nos experimentos foram obtidas da literatura, totalizando 46 instâncias, muitas delas consideradas *benchmark* e disponíveis no website da *EURO Special Interest Group on Cutting and Packing - ESICUP*[1]. Alguns dados dessas instâncias estão na Tabela 2.1, incluindo os autores que as usaram, o número de tipos e o total de itens, as rotações (em graus) permitidas para os itens, as dimensões do recipiente como definidas na literatura e como se propõe neste trabalho, e a escala adotada para a discretização e construção dos *no-fit raster* e *inner-fit raster*. Percebeu-se que para 3 instâncias (ALBANO, MAO, SWIM), uma escala maior (isto é, 1 ponto a cada 10 unidades de distância) é adequada para as heurísticas obterem soluções de qualidade. Nas demais instâncias, a escala adotada foi de 1 ponto a cada 1 unidade de distância.

As primeiras 25 instâncias foram usadas por Mundim *et al.* (2018), incluindo as 15 instâncias definidas em Valle *et al.* (2012). Como elas possuem resultados publicados no contexto do 2IKP, elas são utilizadas na comparação das heurísticas propostas com os melhores métodos da literatura. As demais 21 instâncias não possuem resultados publicados no contexto do 2IKP. Desse modo, objetivando criar um *benchmark* que possa ser utilizado por pesquisadores em trabalhos futuros, propõe-se para todas as 46 instâncias um recipiente em que o comprimento advém de trabalhos da literatura e a largura é definida como o máximo, entre o item de maior largura e a soma da área dos itens dividida pelo comprimento do recipiente, arrendondado para cima. Todas as outras informações sobre essas instâncias permanecem inalteradas, ou seja, elas

[1] https://www.euro-online.org/websites/esicup/data-sets

Tabela 2.1 – Dados das instâncias consideradas para o 2IKP.

Instância	Autores	Total de tipos	Total de itens	Rotações	Lit. (L,W)	Prop. (L,W)	Escala
HAN	Han e Na (1996)	20	23	(0, 90, 180, 270)	(44, 44)	(58, 34)	1:1
ALBANO	Valle *et al.* (2012)	8	24	(0, 180)	(10122,63, 4900)	(8712, 4900)	1:10
DAGLI	Valle *et al.* (2012)	10	30	(0, 180)	(65,60, 60)	(51, 60)	1:1
DIGHE1	Valle *et al.* (2012)	16	16	(0)	(138,13, 100)	(100, 100)	1:1
DIGHE2	Valle *et al.* (2012)	10	10	(0)	(134,50, 100)	(100, 100)	1:1
FU	Valle *et al.* (2012)	11	12	(0, 90, 180, 270)	(34, 38)	(29, 38)	1:1
JAKOBS1	Valle *et al.* (2012)	22	25	(0, 90, 180, 270)	(13, 40)	(10, 40)	1:1
JAKOBS2	Valle *et al.* (2012)	22	25	(0, 90, 180, 270)	(28,20, 70)	(20, 70)	1:1
MAO	Valle *et al.* (2012)	9	20	(0, 90, 180, 270)	(2058,60, 2550)	(1474, 2550)	1:10
MARQUES	Valle *et al.* (2012)	8	24	(0, 180)	(83,60, 104)	(70, 104)	1:1
SHAPES0	Valle *et al.* (2012)	4	43	(0)	(63, 40)	(40, 40)	1:1
SHAPES1	Valle *et al.* (2012)	4	43	(0, 180)	(59, 40)	(40, 40)	1:1
SHAPES2	Valle *et al.* (2012)	7	28	(0, 180)	(27,30, 15)	(22, 15)	1:1
SHIRTS	Valle *et al.* (2012)	8	99	(0, 180)	(63,13, 40)	(54, 40)	1:1
SWIM	Valle *et al.* (2012)	10	48	(0, 180)	(6568, 5752)	(4424, 5752)	1:10
TROUSERS	Valle *et al.* (2012)	17	64	(0, 180)	(245,75, 79)	(218, 79)	1:1
POLY1a	Mundim *et al.* (2018)	15	15	(0, 90, 180, 270)	(20, 20)	(13, 40)	1:1
POLY2a	Mundim *et al.* (2018)	15	30	(0, 90, 180, 270)	(28, 28)	(21, 40)	1:1
POLY2b	Mundim *et al.* (2018)	30	30	(0, 90, 180, 270)	(29, 29)	(23, 40)	1:1
POLY3a	Mundim *et al.* (2018)	15	45	(0, 90, 180, 270)	(34, 34)	(31, 40)	1:1
POLY3b	Mundim *et al.* (2018)	45	45	(0, 90, 180, 270)	(32, 32)	(31, 40)	1:1
POLY4a	Mundim *et al.* (2018)	15	60	(0, 90, 180, 270)	(40, 40)	(41, 40)	1:1
POLY4b	Mundim *et al.* (2018)	60	60	(0, 90, 180, 270)	(39, 39)	(39, 40)	1:1
POLY5a	Mundim *et al.* (2018)	15	75	(0, 90, 180, 270)	(45, 45)	(52, 40)	1:1
POLY5b	Mundim *et al.* (2018)	75	75	(0, 90, 180, 270)	(42, 42)	(46, 40)	1:1
BLASZ2	Oliveira, Gomes e Ferreira (2000)	4	20	(0, 90, 180, 270)	-	(16, 15)	1:1
BLAZEWICZ1	Toledo *et al.* (2013)	7	7	(0, 90, 180, 270)	-	(6, 15)	1:1
BLAZEWICZ2	Toledo *et al.* (2013)	7	14	(0, 90, 180, 270)	-	(11, 15)	1:1
BLAZEWICZ3	Toledo *et al.* (2013)	7	21	(0, 90, 180, 270)	-	(17, 15)	1:1
BLAZEWICZ4	Toledo *et al.* (2013)	7	28	(0, 90, 180, 270)	-	(22, 15)	1:1
BLAZEWICZ5	Toledo *et al.* (2013)	7	35	(0, 90, 180, 270)	-	(27, 15)	1:1
RCO1	Toledo *et al.* (2013)	7	7	(0, 90, 180, 270)	-	(7, 15)	1:1
RCO2	Toledo *et al.* (2013)	7	14	(0, 90, 180, 270)	-	(13, 15)	1:1
RCO3	Toledo *et al.* (2013)	7	21	(0, 90, 180, 270)	-	(19, 15)	1:1
RCO4	Toledo *et al.* (2013)	7	28	(0, 90, 180, 270)	-	(26, 15)	1:1
RCO5	Toledo *et al.* (2013)	7	35	(0, 90, 180, 270)	-	(32, 15)	1:1
SHAPES4	Toledo *et al.* (2013)	4	16	(0, 90, 180, 270)	-	(16, 40)	1:1
SHAPES5	Toledo *et al.* (2013)	4	20	(0, 90, 180, 270)	-	(20, 40)	1:1
SHAPES7	Toledo *et al.* (2013)	4	28	(0, 90, 180, 270)	-	(28, 40)	1:1
SHAPES9	Toledo *et al.* (2013)	4	34	(0, 90, 180, 270)	-	(33, 40)	1:1
SHAPES15	Toledo *et al.* (2013)	4	43	(0, 90, 180, 270)	-	(40, 40)	1:1
THREE	Alvarez-Valdes, Martinez e Tamarit (2013)	3	3	(0, 90, 180, 270)	-	(4, 7)	1:1
THREEp2	Alvarez-Valdes, Martinez e Tamarit (2013)	3	6	(0, 90, 180, 270)	-	(7, 7)	1:1
THREEp2w9	Alvarez-Valdes, Martinez e Tamarit (2013)	3	6	(0, 90, 180, 270)	-	(6, 9)	1:1
THREEp3	Alvarez-Valdes, Martinez e Tamarit (2013)	3	9	(0, 90, 180, 270)	-	(10, 7)	1:1
THREEp3w9	Alvarez-Valdes, Martinez e Tamarit (2013)	3	9	(0, 90, 180, 270)	-	(8, 9)	1:1

são como foram definidas pelos seus autores. Com esse novo recipiente, obtém-se um limitante superior trivial de área ocupada, no intervalo $(0,\ 1]$, que é dado pela soma da área dos itens dividida pela área do recipiente.

2.4.2 Análise das heurísticas

Como forma de validar e avaliar as estruturas de vizinhança da GVNS_{2IKP} sobre as solução obtidas, a Tabela 2.2 apresenta as soluções para as combinações de estruturas (N_1 e N_2). De acordo com Hansen, Mladenović e Pérez (2010), quando apenas uma estrutura de vizinhança é definida na busca local, tem-se uma VNS ao invés da GVNS. Os resultados consideram uma única execução da GVNS_{2IKP} sobre cada uma das 25 instâncias que possuem soluções reportadas na literatura. A coluna "BKS" contém a melhor solução conhecida (*Best-Known Solution*) na literatura para a instância, sendo obtida de Mundim *et al.* (2018), com os melhores resultados até então publicados. As colunas "Dev. (%)" contêm o desvio relativo (isto é, porcentagem de melhora) da solução SOL obtida pela GVNS_{2IKP} com relação a BKS: $100 \times (\text{SOL} - \text{BKS})/\text{BKS}$

(um valor negativo representa o percentual de piora da solução). Valores em negrito indicam as melhores soluções obtidas, isto é, o percentual de melhora sobre a BKS.

Tabela 2.2 – Análise das estruturas de vizinhança da GVNS_{2IKP}.

		$\text{VNS}_{2IKP}(N_1)$		$\text{VNS}_{2IKP}(N_2)$		$\text{GVNS}_{2IKP}(N_1,N_2)$		$\text{GVNS}_{2IKP}(N_2,N_1)$	
Instância	BKS	Dev. (%)	Tempo (s)	Dev. (%)	Tempo (s)	Dev. (%)	Tempo (s)	Dev. (%)	Tempo (s)
HAN	0,7882	5,01	300,00	6,29	300,00	**7,51**	300,00	2,85	300,00
ALBANO	0,8073	**4,12**	300,00	0,00	300,00	**4,12**	300,00	2,55	300,00
DAGLI	0,7710	**0,00**	0,04	**0,00**	0,35	**0,00**	0,17	**0,00**	0,15
DIGHE1	0,7240	**0,00**	0,61	**0,00**	3,14	**0,00**	5,87	**0,00**	4,30
DIGHE2	0,7460	**0,00**	0,13	**0,00**	0,32	**0,00**	0,11	**0,00**	0,46
FU	0,8382	**0,00**	0,03	**0,00**	0,05	**0,00**	0,04	**0,00**	0,22
JAKOBS1	0,7538	**0,00**	0,01	**0,00**	0,01	**0,00**	0,01	**0,00**	0,01
JAKOBS2	0,6844	**0,00**	0,26	**0,00**	0,47	**0,00**	0,01	**0,00**	0,03
MAO	0,7177	**0,00**	0,55	**0,00**	0,87	**0,00**	0,08	**0,00**	0,55
MARQUES	0,8274	**0,00**	0,33	**0,00**	11,24	**0,00**	7,29	**0,00**	8,18
SHAPES0	0,6218	**1,85**	1,58	**1,85**	3,99	**1,85**	3,78	**1,85**	0,62
SHAPES1	0,6496	**4,11**	3,20	**4,11**	10,25	**4,11**	17,14	**4,11**	3,00
SHAPES2	0,7472	2,29	300,00	**5,89**	161,78	**5,89**	35,98	2,29	300,00
SHIRTS	0,8182	**3,29**	300,00	**3,29**	300,00	**3,29**	300,00	**3,29**	300,00
SWIM	0,6610	**0,79**	300,00	**0,79**	300,00	**0,79**	300,00	**0,79**	300,00
TROUSERS	0,7882	**7,74**	300,00	**7,74**	300,00	**7,74**	300,00	**7,74**	300,00
POLY1a	0,7225	5,71	300,00	1,90	300,00	**8,48**	300,00	1,90	300,00
POLY2a	0,6888	9,90	300,00	7,31	300,00	**11,01**	300,00	7,31	300,00
POLY2b	0,6920	8,42	300,00	7,65	300,00	**10,23**	300,00	6,88	300,00
POLY3a	0,6730	**11,96**	300,00	9,45	300,00	**11,96**	300,00	10,80	300,00
POLY3b	0,7148	**6,22**	300,00	4,58	300,00	5,13	300,00	5,13	300,00
POLY4a	0,6606	**11,92**	300,00	**11,92**	300,00	**11,92**	300,00	**11,92**	300,00
POLY4b	0,7068	**7,76**	300,00	2,60	300,00	**7,76**	300,00	6,69	300,00
POLY5a	0,7032	**6,57**	300,00	3,48	300,00	4,32	300,00	3,90	300,00
POLY5b	0,7083	**5,73**	300,00	1,49	300,00	4,89	300,00	4,89	300,00
Média	-	4,14	180,27	3,21	175,70	4,44	170,82	3,40	180,70

Na Tabela 2.2, observa-se que o método proposto obteve igual ou melhor valor de solução para todas as instâncias ao usar apenas a primeira, $\text{VNS}_{2IKP}(N_1)$, ou a segunda, $\text{VNS}_{2IKP}(N_1)$, estrutura, respectivamente, com o aumento percentual médio de 4,14% e 3,21%, e o tempo médio computacional de 180,27 e 175,70 segundos. Ao considerar ambas as estruturas de vizinhança na ordem N_1 e N_2 (ou seja, $\text{GVNS}_{2IKP}(N_1,N_2)$), há uma melhora nos resultados com apenas uma vizinhança, onde o aumento percentual médio é 4,44% e o tempo médio computacional é 170,82 segundos. Por outro lado, se usar ambas as vizinhanças na ordem N_2 e N_1 (ou seja, $\text{GVNS}_{2IKP}(N_2,N_1)$), o aumento percentual médio é 3,40 % e a média computacional do tempo é 180,70 segundos, o que é pior do que se usar apenas a vizinhança N_1 (ou seja, $\text{VNS}_{2IKP}(N_1)$).

Nota-se que com as vizinhanças na ordem N_1 e depois N_2, melhores soluções são obtidas para 22 das 25 instâncias. Com apenas N_1 este número é de 19, com apenas N_2 é de 15 e com N_2 e depois N_1 é de 14 instâncias. Todos esses resultados indicam que a melhor escolha é considerar as duas estruturas de vizinhança na ordem N_1 (troca) e depois N_2 (inserção).

Outra análise é feita para verificar quão rápidas as heurísticas BRKGA_{2IKP} e GVNS_{2IKP} convergem para uma dada solução. Apresentam-se as distribuições empíricas do tempo que as heurísticas requerem para obter uma solução alvo (AIEX; BINATO; RESENDE, 2003). Estima-se a probabilidade de obter uma solução tão boa quanto a solução alvo para um dado tempo

execução. Para a construção dos gráficos com as distribuições empíricas de tempo, consideram-se os tempos de execução t_j, em ordem crescente, sobre o eixo x, e as probabilidades $p_j = \frac{j-0,5}{w}$ associada ao tempo t_j, resultando no ponto (t_j, p_j), para $j = 1,2,\ldots,w$. Nos experimentos, adotou-se $w = 100$ (para diferentes sementes), o tempo limite de 300 segundos e a solução alvo como sendo a BKS informada na Tabela 2.2.

A Figura 2.4 contém as distribuições empíricas para quatro instâncias da Tabela 2.2, entre pequenas, médias e grandes, que foram escolhidas para exemplificar o desempenho dos métodos propostos. Em todos os casos, o BRKGA$_{2IKP}$ é melhor que a GVNS$_{2IKP}$, ou seja, com maior probabilidade de atingir a solução alvo (BKS) em menor tempo computacional. Além disso, o BRKGA$_{2IKP}$ pode atingir a BKS para todas essas instâncias dentro do limite de tempo de 300 segundos, o que pode ser explicado pelo BRKGA$_{2IKP}$ trabalhar sobre uma população de indivíduos. Nas instâncias, HAN e POLY5b, o BRKGA$_{2IKP}$ tem uma probabilidade próxima a 1 para um tempo de execução inferior a 30 segundos. Para a GVNS$_{2IKP}$, isso acontece para um tempo de execução entre 200 e 300 segundos. Para a instância SWIM, o BRKGA$_{2IKP}$ e a GVNS$_{2IKP}$ têm uma probabilidade próxima de 1 para atingir a BKS com o tempo entre 140 e 180 segundos.

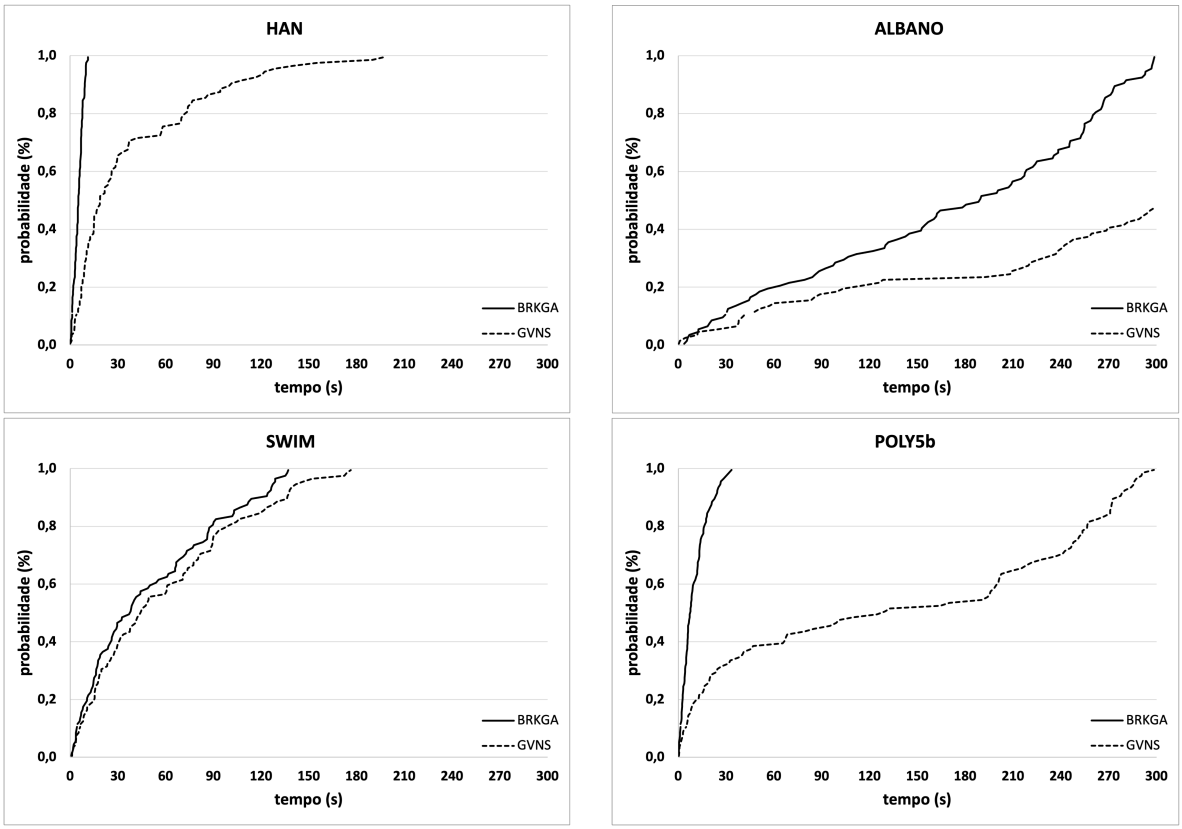


Figura 2.4 – Distribuições empíricas de tempo para algumas instâncias.

Para a instância ALBANO na Figura 2.4, o BRKGA$_{2IKP}$ requer mais tempo de execução ao comparar sua execução nas outras instâncias para atingir a BKS com alta probabilidade. Percebe-se que à medida que a probabilidade se aproxima de 1 para atingir a BKS, seu tempo

computacional se aproxima dos 300 segundos. Por outro lado, a GVNS$_{2IKP}$ não é capaz de obter a BKS com alta probabilidade para a ALBANO. Ela atinge o limite de tempo de 300 segundos com uma probabilidade próxima de 0,5, o que significa que sua probabilidade de obter a BKS em 300 segundos é menor que 50% dadas 100 execuções com diferentes sementes.

2.4.3 Comparação com a literatura

Os resultados obtidos pelos métodos propostos são comparados com resultados da literatura. Para isso, utilizam-se as primeiras 25 instâncias da Tabela 2.1, com as dimensões do recipiente conforme definidas na literatura (coluna "Lit. (L,W)"). É feita uma comparação com o método H4NP de Mundim *et al.* (2018), que, até onde se conhece, resolveu todas essas 25 instâncias, apresentando os melhores resultados até o momento em termos de solução e tempo computacional. Vale ressaltar que esses autores compararam seu método com os melhores métodos publicados anteriormente na literatura, incluindo o recozimento simulado de Gomes e Oliveira (2006) e Martins e Tsuzuki (2010), o GRASP de Valle *et al.* (2012), a heurística construtiva de Dalalah, Khrais e Bataineh (2014), e os modelos de programação inteira de Fischetti e Luzzi (2009) e Alvarez-Valdes, Martinez e Tamarit (2013).

Para uma comparação justa com Mundim *et al.* (2018), seu método foi executado no mesmo computador em que se executou as heurísticas propostas, com cada instância sendo resolvida 10 vezes (para diferentes sementes). O método deles considerou os mesmos critérios de parada que se adotam para o BRKGA$_{2IKP}$ e a GVNS$_{2IKP}$ (ou seja, parar após atingir um limite de tempo de 300 segundos ou, antes desse tempo, ao obter uma solução ótima conhecida). A Tabela 2.3 apresenta os resultados em termos do pior, médio e melhor valor da solução (em termos de área ocupada, para as 10 execuções) e o tempo médio computacional (em segundos). Para os métodos propostos, também é apresentado o desvio percentual de sua melhor solução em relação à melhor solução reportada por Mundim *et al.* (2018). Os valores marcados com um '*' indicam que o padrão de corte é ótimo para o 2IKP.

Os resultados da Tabela 2.3 mostram que os métodos propostos encontraram valores iguais (para 8 instâncias) ou melhores (para 17 instâncias), observando a melhor solução, comparados com a melhor solução do método de Mundim *et al.* (2018). Em termos percentuais, o BRKGA$_{2IKP}$ aumentou a área ocupada em cerca de 6,94% sobre a literatura, enquanto a GVNS$_{2IKP}$ obteve uma melhoria de 5,93%. A instância para a qual os métodos propostos mais melhoraram a área ocupada é a POLY3a, com uma melhoria de 17,68%, para o BRKGA$_{2IKP}$, e de 17,16%, para o GVNS$_{2IKP}$. Vale ressaltar que o BRKGA$_{2IKP}$ atingiu um valor ótimo da solução para 15 instâncias, enquanto a GVNS$_{2IKP}$ fez o mesmo para 14 instâncias. Esses valores são quase o dobro do número reportado pelo H4NP de Mundim *et al.* (2018). A Figura 2.5 apresenta as soluções com a área mais bem ocupada que são relatadas na Tabela 2.3. As soluções de (a)-(w) foram obtidas com o BRKGA$_{2IKP}$, enquanto as duas últimas com a GVNS$_{2IKP}$.

Comparando os métodos propostos utilizando os resultados da Tabela 2.3, o BRKGA$_{2IKP}$

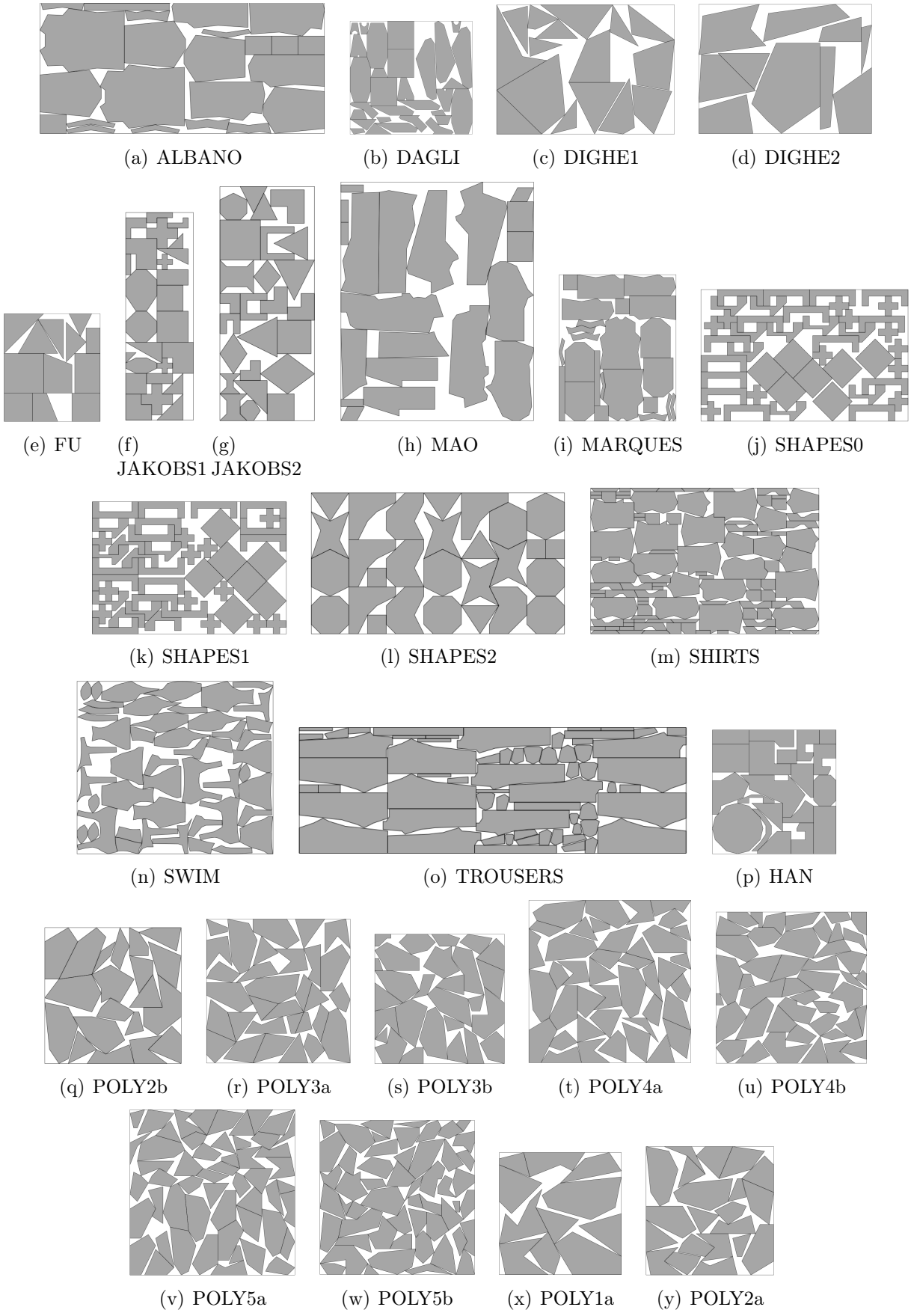
Tabela 2.3 – Comparação das heurísticas em instâncias da literatura.

Instância	H4NP de Mundim *et al.* (2018)				BRKGA$_{2IKP}$					GVNS$_{2IKP}$				
	Pior	Média	Melhor	*Tempo* (*s*)	Pior	Média	Melhor	Dev. (%)	*Tempo* (*s*)	Pior	Média	Melhor	Dev. (%)	*Tempo* (*s*)
HAN	0,7335	0,7423	0,7882	300,00	0,8551	0,8604	0,8768	**11,24**	300,00	0,8572	0,8592	0,8678	10,10	300,00
ALBANO	0,7650	0,7842	0,8073	300,00	0,8371	0,8433	0,8620*	**6,78**	300,00	0,8336	0,8383	0,8620*	**6,78**	300,00
DAGLI	0,7710*	0,7710*	0,7710*	45,42	0,7710*	0,7710*	0,7710*	**0,00**	0,55	0,7710*	0,7710*	0,7710*	**0,00**	5,53
DIGHE1	0,7063	0,7144	0,7240*	240,83	0,7240*	0,7240*	0,7240*	**0,00**	7,58	0,7240*	0,7240*	0,7240*	**0,00**	21,15
DIGHE2	0,7460*	0,7460*	0,7460*	23,19	0,7460*	0,7460*	0,7460*	**0,00**	0,44	0,7460*	0,7460*	0,7460*	**0,00**	0,84
FU	0,8382*	0,8382*	0,8382*	36,62	0,8382*	0,8382*	0,8382*	**0,00**	0,46	0,8382*	0,8382*	0,8382*	**0,00**	0,89
JAKOBS1	0,7423	0,7492	0,7538*	215,12	0,7538*	0,7538*	0,7538*	**0,00**	0,01	0,7538*	0,7538*	0,7538*	**0,00**	0,01
JAKOBS2	0,6707	0,6721	0,6844*	271,04	0,6844*	0,6844*	0,6844*	**0,00**	0,03	0,6844*	0,6844*	0,6844*	**0,00**	0,04
MAO	0,7009	0,7101	0,7177*	300,00	0,7177*	0,7177*	0,7177*	**0,00**	0,67	0,7177*	0,7177*	0,7177*	**0,00**	2,32
MARQUES	0,8182	0,8191	0,8274*	300,00	0,8274*	0,8274*	0,8274*	**0,00**	6,26	0,8274*	0,8274*	0,8274*	**0,00**	35,21
SHAPES0	0,5492	0,5595	0,6218	300,00	0,6333*	0,6333*	0,6333*	**1,85**	3,49	0,6333*	0,6333*	0,6333*	**1,85**	37,47
SHAPES1	0,5525	0,5648	0,6496	300,00	0,6763*	0,6763*	0,6763*	**4,11**	3,42	0,6763*	0,6763*	0,6763*	**4,11**	27,96
SHAPES2	0,7204	0,7231	0,7472	300,00	0,7912*	0,7912*	0,7912*	**5,89**	52,18	0,7912*	0,7912*	0,7912*	**5,89**	91,00
SHIRTS	0,8039	0,8069	0,8182	300,00	0,8554*	0,8554*	0,8554*	**4,54**	34,77	0,8554*	0,8554*	0,8554*	**4,54**	201,63
SWIM	0,5604	0,5728	0,6610	300,00	0,6748*	0,6748*	0,6748*	**2,09**	59,64	0,6748*	0,6748*	0,6748*	**2,09**	115,37
TROUSERS	0,6887	0,7010	0,7882	300,00	0,8812	0,8830	0,8863*	**12,45**	300,00	0,8492	0,8531	0,8796	11,60	300,00
POLY1a	0,7225	0,7225	0,7225	300,00	0,7775	0,7905	0,8013	10,90	300,00	0,8025	0,8025	0,8025	**11,07**	300,00
POLY2a	0,6888	0,6888	0,6888	300,00	0,7832	0,7878	0,7959	15,55	300,00	0,7742	0,7790	0,7978	**15,82**	300,00
POLY2b	0,6920	0,6920	0,6920	300,00	0,7806	0,7866	0,7949	**14,87**	300,00	0,7717	0,7771	0,7931	14,61	300,00
POLY3a	0,6730	0,6730	0,6730	300,00	0,7721	0,7772	0,7920	**17,68**	300,00	0,7612	0,7679	0,7885	17,16	300,00
POLY3b	0,6982	0,7027	0,7148	300,00	0,7944	0,7999	0,8130	**13,74**	300,00	0,7695	0,7751	0,7808	9,23	300,00
POLY4a	0,6606	0,6606	0,6606	300,00	0,7650	0,7688	0,7766	**17,56**	300,00	0,7441	0,7486	0,7728	16,98	300,00
POLY4b	0,6798	0,6848	0,7068	300,00	0,7679	0,7740	0,7972	**12,79**	300,00	0,7390	0,7463	0,7528	6,51	300,00
POLY5a	0,6943	0,6988	0,7032	300,00	0,7578	0,7632	0,7741	**10,08**	300,00	0,7336	0,7359	0,7405	5,30	300,00
POLY5b	0,6780	0,6827	0,7083	300,00	0,7605	0,7658	0,7888	**11,37**	300,00	0,7344	0,7370	0,7418	4,73	300,00
Média	-	-	-	261,29	-	-	-	6,94	150,78	-	-	-	5,93	165,58

obteve a melhor área ocupada para 9 instâncias (HAN, TROUSERS, POLY2b, POLY3a, POLY3b, POLY4a, POLY4b, POLY5a e POLY5b), com o aumento percentual de 2,65% sobre a GVNS$_{2IKP}$ nestes casos. Por outro lado, a GVNS$_{2IKP}$ teve a área mais bem ocupada para 2 instâncias (POLY1a e POLY2a), com o aumento percentual de 0,20% sobre o BRKGA$_{2IKP}$ em tais instâncias. No geral, o BRKGA$_{2IKP}$ obteve valores da solução melhores em comparação com a GVNS$_{2IKP}$, mas ambas as heurísticas são muito superiores ao melhor método da literatura. Observe que ao longo das 10 execuções, o método de Mundim *et al.* (2018) obteve as mesmas soluções piores, médias e melhores para apenas 3 instâncias (DAGLI, DIGHE2 e FU), enquanto o BRKGA$_{2IKP}$ e a GVNS$_{2IKP}$ obtiveram o mesmo para 13 instâncias (DAGLI, DIGHE1, DIGHE2, FU, JAKOBS1, JAKOBS2, MAO, MARQUES, SHAPES0, SHAPES1, SHAPES2, SHIRTS e SWIM).

Em relação ao tempo computacional médio, o H4NP de Mundim *et al.* (2018) teve o maior, com média geral de 261,29 segundos e atingindo o limite de tempo para 19 das 25 instâncias. Em relação ao tempo de execução do BRKGA$_{2IKP}$, a média geral é de 150,78 segundos, atingindo o limite de tempo para 12 instâncias. A GVNS$_{2IKP}$ tem o segundo maior tempo de execução, com uma média geral de 165,58 segundos, atingindo o limite de tempo para 12 instâncias. O maior tempo de execução da GVNS$_{2IKP}$ em comparação com o BRKGA$_{2IKP}$ pode ser justificado pela fase de busca local, que testa todos os movimentos e, para cada movimento, até três regras de posicionamento podem ser executadas.

Para comparar os métodos da Tabela 2.3 em termos de suas soluções nas 25 instâncias, fez-se o teste estatístico de postos sinalizados de Wilcoxon. Derrac *et al.* (2011) comentaram que este teste verifica se existe uma diferença no comportamento de dois métodos sobre um dado conjunto de instâncias. Foi utilizado o *software* estatístico R, com correção de continuidade e exclusão de *outliers*, para o nível de significância (α) de 1% (ou seja, um intervalo de confiança



(a) ALBANO (b) DAGLI (c) DIGHE1 (d) DIGHE2

(e) FU (f) JAKOBS1 (g) JAKOBS2 (h) MAO (i) MARQUES (j) SHAPES0

(k) SHAPES1 (l) SHAPES2 (m) SHIRTS

(n) SWIM (o) TROUSERS (p) HAN

(q) POLY2b (r) POLY3a (s) POLY3b (t) POLY4a (u) POLY4b

(v) POLY5a (w) POLY5b (x) POLY1a (y) POLY2a

Figura 2.5 – Melhores soluções obtidas pelas heurísticas propostas, conforme a Tabela 2.3.

de 99%). O teste considera as soluções nas colunas “Média”, onde a hipótese alternativa é a unilateral à esquerda, indicando que a mediana das soluções do primeiro método é pior do que a do segundo método. Portanto, o resultado do teste estatístico para os pares de métodos (H4NP, BRKGA$_{2IKP}$) e (H4NP, GVNS$_{2IKP}$) indicam a rejeição da hipótese nula e aceitação da hipótese alternativa, com o p-valor igual a 0,000000238. Isso confirma que há significância estatística entre os resultados do método de Mundim *et al.* (2018) e os dos métodos propostos. Finalmente, o resultado para o par (GVNS$_{2IKP}$, BRKGA$_{2IKP}$) retorna um p-valor de 0,003418, indicando também a aceitação da hipótese alternativa.

2.4.4 Novos resultados

Com o objetivo de contribuir com a literatura em termos de instâncias e resultados que os pesquisadores possam utilizar em futuras comparações, apresentam-se resultados para as 46 instâncias da Tabela 2.1 considerando as dimensões do recipiente, conforme se propõe na coluna “Prop. (L,W)”. Em relação ao recipiente proposto para cada instância, seu comprimento é o definida na literatura, enquanto sua largura é o máximo, arredondado para cima, entre o item de maior largura e a soma da largura dos itens dividida pelo comprimento do recipiente. Para cada método, os resultados são apresentados, dadas as 10 execuções por instância, em termos da pior, média e melhor solução, e a diferença percentual em relação ao limite superior (quanto menor, mais próximo de zero, é melhor). Os valores marcados com um ‘*’ indicam que o padrão de corte é ótimo.

Os novos resultados são fornecidos na Tabela 2.4. Cada linha desta tabela tem o nome da instância, o limite superior trivial (na coluna “UB”), a pior, a média e a melhor solução (respectivamente, nas colunas “Pior”, “Média” e “Melhor”), o desvio percentual entre a melhor solução e o limite superior (nas colunas “GAP (%)”), e a iteração (geração) onde a melhor solução foi encontrada. A coluna com o tempo computacional não é mostrada porque os métodos propostos só pararam após atingir os 300 segundos, para todas as 46 instâncias, exceto duas delas (DIGHE1 e DIGHE2).

Na Tabela 2.4, em relação às melhores soluções, o BRKGA$_{2IKP}$ obteve resultados iguais, melhores e piores para, respectivamente, 19, 16 e 11 das 46 instâncias, em comparação com a GVNS$_{2IKP}$. Dentre as instâncias em que o BRKGA$_{2IKP}$ obteve melhores resultados, o desvio percentual mais significativo é de 6,25% para a instância SHAPES4, enquanto o desvio percentual médio é 2,12%. Dentre as instâncias onde o BRKGA$_{2IKP}$ teve resultados piores, o desvio percentual mais substancial é 1,89%, para a instância POLY2a, enquanto o desvio percentual médio é 0,55%. Em relação ao GAP médio, o BRKGA$_{2IKP}$ superou a GVNS$_{2IKP}$, com um valor de 19,80% contra 20,58%. Em termos do pior GAP, o BRKGA$_{2IKP}$ obteve 41,99% para a instância SHAPES0, enquanto a GVNS$_{2IKP}$ obteve 42,86% para a instância SHAPES4. Ambos os métodos alcançaram a solução ótima conhecida para duas instâncias (DIGHE1 e DIGHE2), com 100% da área ocupada.

Tabela 2.4 – Resultados das heurísticas propostas sobre as instâncias adaptadas.

		BRKGA$_{2IKP}$					GVNS$_{2IKP}$				
Instância	UB	Pior	Média	Melhor	GAP (%)	Iter.	Pior	Média	Melhor	GAP (%)	Iter.
HAN	0,9990	0,8486	0,8535	0,8638	15,65	152	0,8453	0,8548	0,8669	**15,24**	7
ALBANO	0,9999	0,8315	0,8390	0,8546	**17,00**	5	0,8248	0,8299	0,8457	18,23	1
DAGLI	0,9917	0,8753	0,8776	0,8837	12,23	1937	0,8690	0,8738	0,8868	**11,83**	9
DIGHE1	1,0000	1,0000*	1,0000*	1,0000*	**0,00**	12	1,0000*	1,0000*	1,0000*	**0,00**	2
DIGHE2	1,0000	1,0000*	1,0000*	1,0000*	**0,00**	8	1,0000*	1,0000*	1,0000*	**0,00**	2
FU	0,9828	0,9256	0,9256	0,9256	**6,18**	15	0,9256	0,9256	0,9256	**6,18**	5
JAKOBS1	0,9800	0,9200	0,9210	0,9250	5,95	1985	0,9200	0,9276	0,9338	**4,95**	21
JAKOBS2	0,9650	0,8421	0,8478	0,8636	11,75	699	0,8507	0,8566	0,8664	**11,38**	6
MAO	0,9999	0,8175	0,8208	0,8280	**20,76**	13	0,8140	0,8167	0,8271	20,89	1
MARQUES	0,9882	0,8846	0,8861	0,8904	10,99	63	0,8775	0,8823	0,8951	**10,40**	9
SHAPES0	0,9975	0,7025	0,7028	0,7050	41,49	604	0,7025	0,7038	0,7075	**40,99**	2
SHAPES1	0,9975	0,7350	0,7368	0,7425	34,34	1335	0,7325	0,7370	0,7450	**33,89**	5
SHAPES2	0,9818	0,8485	0,8485	0,8485	**15,71**	35	0,8485	0,8485	0,8485	**15,71**	9
SHIRTS	1,0000	0,8870	0,8895	0,8942	**11,83**	310	0,8718	0,8742	0,8824	13,33	1
SWIM	0,9999	0,7164	0,7198	0,7364	**35,78**	5	0,7024	0,7073	0,7235	38,21	1
TROUSERS	0,9991	0,8891	0,8919	0,9004	**10,96**	93	0,8593	0,8640	0,8768	13,95	1
POLY1a	0,7885	0,7519	0,7556	0,7673	**2,76**	2451	0,7654	0,7669	0,7673	**2,76**	16
POLY2a	0,9762	0,7667	0,7716	0,7869	24,06	637	0,7702	0,7776	0,8018	**21,75**	14
POLY2b	0,9832	0,7701	0,7761	0,7848	25,28	65	0,7614	0,7674	0,7864	**25,03**	2
POLY3a	0,9919	0,7641	0,7708	0,7855	**26,28**	293	0,7484	0,7533	0,7633	29,95	1
POLY3b	0,9839	0,7718	0,7777	0,7879	**24,88**	225	0,7504	0,7587	0,7851	25,32	1
POLY4a	1,0000	0,7588	0,7641	0,7716	**29,59**	157	0,7418	0,7470	0,7543	32,58	1
POLY4b	0,9917	0,7609	0,7667	0,7981	**24,26**	65	0,7385	0,7436	0,7622	30,11	1
POLY5a	0,9856	0,7514	0,7563	0,7654	**28,77**	254	0,7320	0,7371	0,7548	30,58	1
POLY5b	0,9978	0,7481	0,7548	0,7769	**28,43**	50	0,7228	0,7275	0,7505	32,95	1
BLASZ2	0,9417	0,7917	0,7917	0,7917	**18,95**	86	0,7917	0,7917	0,7917	**18,95**	1
BLAZEWICZ1	0,9000	0,7778	0,7778	0,7778	**15,71**	2	0,7778	0,7778	0,7778	**15,71**	1
BLAZEWICZ2	0,9818	0,8485	0,8485	0,8485	**15,71**	9	0,8485	0,8485	0,8485	**15,71**	5
BLAZEWICZ3	0,9529	0,8627	0,8651	0,8667	**9,95**	1209	0,8667	0,8667	0,8667	**9,95**	11
BLAZEWICZ4	0,9818	0,8515	0,8617	0,8697	12,89	121	0,8561	0,8632	0,8727	**12,50**	35
BLAZEWICZ5	1,0000	0,8568	0,8632	0,8765	14,08	2201	0,8642	0,8668	0,8802	**13,60**	25
RCO1	0,9000	0,7667	0,7667	0,7667	**17,39**	2	0,7667	0,7667	0,7667	**17,39**	1
RCO2	0,9692	0,9077	0,9169	0,9179	**5,58**	848	0,9077	0,9077	0,9077	6,78	14
RCO3	0,9947	0,9175	0,9196	0,9246	**7,59**	741	0,9175	0,9232	0,9246	**7,59**	14
RCO4	0,9692	0,9128	0,9133	0,9179	**5,58**	1342	0,9128	0,9133	0,9179	**5,58**	42
RCO5	0,9844	0,9073	0,9106	0,9177	**7,27**	209	0,9094	0,9135	0,9177	**7,27**	10
SHAPES4	1,0000	0,7438	0,7438	0,7438	**34,45**	8	0,7000	0,7000	0,7000	42,86	2
SHAPES5	1,0000	0,7250	0,7260	0,7300	**36,99**	476	0,7200	0,7205	0,7250	37,93	2
SHAPES7	1,0000	0,7571	0,7643	0,7750	**29,03**	1047	0,7536	0,7600	0,7750	**29,03**	9
SHAPES9	0,9818	0,7576	0,7639	0,7758	**26,56**	377	0,7515	0,7542	0,7576	29,60	2
SHAPES15	0,9975	0,7475	0,7513	0,7650	**30,39**	1120	0,7400	0,7453	0,7550	32,12	5
THREE	0,8214	0,6071	0,6071	0,6071	**35,29**	2	0,6071	0,6071	0,6071	**35,29**	1
THREEp2	0,9388	0,6939	0,6939	0,6939	**35,30**	2	0,6939	0,6939	0,6939	**35,30**	1
THREEp2w9	0,8519	0,7037	0,7037	0,7037	**21,06**	2	0,7037	0,7037	0,7037	**21,06**	1
THREEp3	0,9857	0,7000	0,7000	0,7000	**40,81**	2	0,7000	0,7000	0,7000	**40,81**	1
THREEp3w9	0,9583	0,7639	0,7639	0,7639	**25,45**	2	0,7639	0,7639	0,7639	**25,45**	3
Média	-	-	-	-	19,80	463	-	-	-	20,58	7

O número de gerações/iterações depende do tempo que cada método requer para executar cada geração. Como a GVNS$_{2IKP}$ reinicia a busca para a primeira estrutura de vizinhança quando a solução atual é melhorada, em geral, esta heurística pode requerer mais tempo por iteração. Então, observando a Tabela 2.4, o BRKGA$_{2IKP}$ obteve o melhor valor de solução em 100 ou menos gerações para 22 instâncias, enquanto a GVNS$_{2IKP}$ fez o mesmo para todas as 46 instâncias.

As piores soluções e as soluções médias relatadas na Tabela 2.4, dadas as 10 execuções de cada método, são iguais às melhores soluções para 14 instâncias, para o BRKGA$_{2IKP}$, e 16 instâncias, para a GVNS$_{2IKP}$. Para as demais instâncias, os maiores desvios percentuais entre a média e as melhores soluções são 3,94% (para POLY4b) e 3,36% (para POLY3b), com uma média de 1,24% e 1,41%, respectivamente. Em relação ao tempo computacional, ambas as heurísticas pararam somente após atingir o limite de tempo de 300 segundos, exceto para as instâncias DIGHE1 e DIGHE2, cujo tempo médio de execução geral foi de 8,31 segundos para resolver cada uma delas.

Com relação ao teste de postos sinalizados de Wilcoxon aplicado sobre as heurísticas e seus resultados na Tabela 2.4, consideram-se as soluções das colunas "Média", onde a hipótese alternativa indica que a mediana das soluções do primeiro método é pior do que o do segundo método. Portanto, o resultado do teste estatístico para o par de métodos (GVNS$_{2IKP}$, BRKGA$_{2IKP}$) indica a rejeição da hipótese nula e aceitação da hipótese alternativa, com o p-valor de 0,0074. Isso confirma que há significância estatística entre os métodos.

A Figura 2.6 mostra o número de vezes que cada regra de posicionamento permitiu uma solução melhor do que a corrente. Esses números foram obtidos a partir da execução que resultou na melhor solução da Tabela 2.4, para cada uma das 46 instâncias. No BRKGA$_{2IKP}$ e na GVNS$_{2IKP}$, a regra *horizontal zig-zag alternada* foi a melhor, seguida pela regra *bottom-left*. Em geral, a regra *horizontal zig-zag alternada* permitiu obter o maior número de melhores soluções, totalizando 527, contra a regra *bottom-left*, a segunda com um total de 452. Portanto, embora a regra *bottom-left*, amplamente utilizada no literatura, possa fornecer boas soluções, outras regras de posicionamento são necessárias se soluções de alta qualidade são desejadas.

2.5 Considerações finais

O problema de corte de itens irregulares foi resolvido por um algoritmo genético de chaves aleatórias viciadas, denominado BRKGA$_{2IKP}$, e por uma busca em vizinhança variável geral, denominada GVNS$_{2IKP}$. Nas heurísticas propostas, o vetor carrega informações sobre a sequência e a rotação em que os itens devem ser alocados no recipiente, além da quantidade de posições viáveis que devem ser puladas para posicionar itens. No BRKGA$_{2IKP}$, o vetor ainda carrega a probabilidade, caso o cromossomo seja elite, de suas informações serem passadas para os descendentes. Além da clássica regra de posicionamento *bottom-left*, duas outras foram

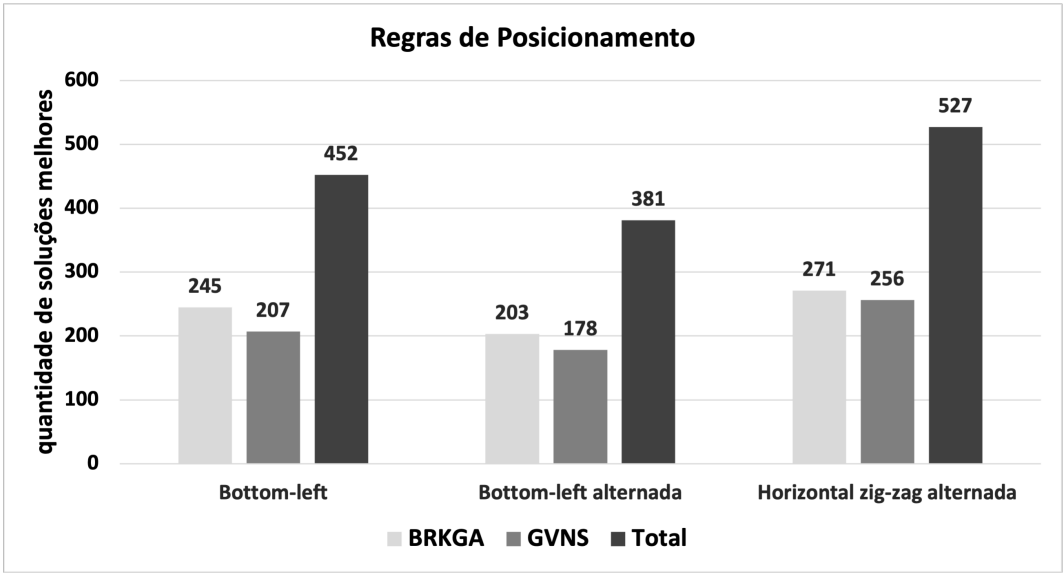


Figura 2.6 – Número de melhores soluções obtidas com cada regra de posicionamento.

propostas, permitindo trazer bons resultados para o problema.

Na comparação com a literatura, as heurísticas foram iguais ou superiores em todas as instâncias, sendo possível obter a solução ótima para 60% das instâncias. Além disso, as heurísticas conseguiram soluções cuja área ocupada do recipiente aumentou em média 6,44%. Em algumas instâncias foi possível aumentar a área ocupada em 17,68%, em média, o que representa um ganho bastante significativo. No geral, o BRKGA$_{2IKP}$ é capaz de obter boas soluções mais rapidamente, por considerar uma população de indivíduos e escapar de ótimos locais com a introdução de indivíduos mutantes e uma probabilidade de cruzamento que não é constante mas faz parte do indivíduo. Também, o teste dos postos sinalizados de Wilcoxon mostrou que os resultados das heurísticas são estatisticamente significativos entre si para um intervalo de confiança de 99%.

CAPÍTULO

3

ALGORITMO *BRANCH-AND-CUT* PARA O PROBLEMA DE CORTE EM FAIXA COM INCERTEZAS NAS DEMANDAS

Neste capítulo, apresenta-se um algoritmo *branch-and-cut* para o problema de corte em faixa de itens irregulares bidimensionais com incertezas sobre a demanda dos itens a serem cortados. Desenvolve-se um modelo de programação estocástica de dois estágios, considerando um conjunto discreto e finito de cenários. No modelo, a faixa é discretizada sobre uma malha de pontos e inclui restrições para assegurar a viabilidade do empacotamento baseadas nos conceitos de *inner-fit raster* e *no-fit raster*. O algoritmo considera o cálculo de limitantes inferiores rápidos e limitantes superiores a partir de uma heurística baseada na busca em vizinhança variável. A heurística também é utilizada durante a otimização para podar nós insatisfatórios. Os resultados numéricos indicam a eficácia do algoritmo proposto ao observar outros algoritmos exatos no mesmo problema sem incerteza. O algoritmo também pode fornecer soluções ótimas para instâncias com incertezas contendo mais de 60 cenários dentro de algumas horas de execução. Além disso, as conclusões mostram que é preferível lidar com as incertezas para obter soluções de menor custo.

Os resultados iniciais desse capítulo foram publicados na forma de um artigo completo e apresentados no *LII Simpósio Brasileiro de Pesquisa Operacional* (SOUZA QUEIROZ; ANDRETTA, 2020a). Todo o capítulo foi publicado no periódico *International Transactions in Operational Research* (SOUZA QUEIROZ; ANDRETTA, 2022).

3.1 Introdução

Vários problemas que surgem no contexto real têm sido resolvidos sob a hipótese de que os dados são determinísticos e conhecidos *a priori*. Todavia, essa suposição tem levado, em

muitos casos, a métodos e/ou soluções que não representam adequadamente a realidade, já que os dados podem estar sujeitos a incertezas por diversas razões, como não existirem no momento em que o problema é resolvido, influência de fatores externos (mercado ou ações), ou não foram coletados ou medidos exatamente (SOYSTER, 1973).

Problemas de corte e empacotamento estão relacionados com o corte de objetos ou o empacotamento de itens buscando otimizar algum objetivo. Esses problemas são de grande importância prática, sendo comumente encontrados em empresas e indústrias, como na metal-mecânica, na automotiva e de máquinas agrícolas, na têxtil, na de produção de móveis, e no transporte logístico e fretamento (WÄSCHER; HAUSSNER; SCHUMANN, 2007). Esses problemas também estão sujeitos a incertezas, que podem surgir, por exemplo, no valor, no tamanho, no peso, na demanda e na qualidade dos itens e recipientes (CRAINIC *et al.*, 2016; SARPER; JAKSIC, 2019).

Problemas que envolvem o corte e empacotamento de itens irregulares possuem a dificuldade adicional relacionada à geometria dos itens, o que dificulta na proposta de métodos de resolução que sejam eficazes. Um tutorial de como lidar com a questão geométrica, em especial quando se deseja verificar sobreposição entre os itens, foi feito por Bennell e Oliveira (2008). Esses autores destacaram os métodos: *raster*, que utiliza uma matriz de pontos para representar os itens e recipientes; *phi-function*, que verifica a posição relativa entre os itens a partir de funções de distância; trigonometria direta, que verifica a interseção entre segmentos de retas e inclusão de pontos; e *no-fit polygon*, que é o polígono obtido ao transladar um item ao redor de outro e traz a representação de onde os itens estão encostando, sobrepostos ou não. Toledo *et al.* (2013) fez a combinação dos métodos *raster* e *no-fit polygon* para obter o *no-fit raster*.

Neste trabalho, investiga-se um problema corte de itens irregulares em que é preciso decidir qual o tamanho de área de faixa retangular a comprar a fim de produzir itens demandados. A área da faixa é determinada pelos itens que vão ser cortados, cuja demanda é desconhecida inicialmente, sendo precisamente revelada em um tempo futuro. Como a faixa requer um determinado tempo de preparação no fornecedor, o preço por metro quadrado varia conforme o momento (inicial ou futuro) em que a área da faixa é encomendada. Se mais área de faixa é solicitada no futuro, essa metragem adicional custa (bem) mais caro do que quando solicitada inicialmente. O objetivo é decidir quantos metros de área de faixa solicitar (inicialmente e, se for o caso, no futuro) observando uma previsão de demanda dos itens, minimizando o custo total da faixa. Denomina-se esse problema de Problema de Corte de itens Irregulares em Faixa Bidimensional com Demanda Incerta (*Two-Dimensional Irregular Strip Packing Problem with Demand Uncertainty - 2ISP-DU*) . Para este problema, desenvolve-se um algoritmo exato *branch-and-cut* que parte da resolução de um modelo de programação estocástica de dois estágios com recurso, considera o *no-fit raster* para as questões de não sobreposição entre itens e utiliza uma heurística para também fornecer limitantes válidos.

3.1.1 Revisão da literatura

A versão determinística do 2ISP está na classe NP-Difícil (detalhes sobre as classes de complexidade são dados por Garey e Johnson (1979)) e tem sido resolvida por diferentes algoritmos, em sua maioria heurísticas. Albano e Sapuppo (1980) buscavam, dada uma sequência de itens, o melhor item que poderia ser empacotado por uma heurística *bottom-left*. Jakobs (1996) desenvolveu um algoritmo genético em que cada indivíduo representa uma sequência para empacotar os itens. Os itens são empacotados por uma heurística *bottom-left*. Gomes e Oliveira (2002) trabalharam sobre uma sequência de itens, com as posições iniciais válidas para empacotar os itens sendo obtidas do *inner-fit polygon* e do *no-fit polygon*. Critérios foram adotados para gerar sequências para empacotar os itens por uma heurística *bottom-left*. Uma busca local foi usada para trocar itens de posição nas sequências.

Em Gomes e Oliveira (2006) há um recozimento simulado, com a heurística *bottom-left* para posicionar os itens, um modelo de programação linear para remover sobreposições e outro modelo de programação linear para realizar a compactação da solução. Uma heurística híbrida que combina *cuckoo search* e busca local guiada foi proposta por Elkeran (2013). O autor considerou o agrupamento de itens em uma fase de pré-processamento, enquanto a heurística utiliza o *no-fit polygon* para lidar com a sobreposição entre itens. Mundim, Andretta e Queiroz (2017), por outro lado, desenvolveram um algoritmo genético de chaves aleatórias viciadas, que utiliza a heurística *bottom-left* para o posicionamento de itens. Nesse algoritmo, o cromossomo indica a sequência e a rotação para empacotar os itens.

As propostas de algoritmos exatos para o 2ISP têm considerado em sua maioria a resolução de modelos matemáticos. Carravilla *et al.* (2003) propuseram modelos resolvidos por programação por restrições, que consideram itens convexos e não convexos. A não sobreposição dos itens é verificada pelo *no-fit polygon* e há restrições para remover soluções simétricas. Em Fischetti e Luzzi (2009) há um modelo de programação linear inteira mista, que define a posição dos itens por meio de variáveis contínuas. Para gerar as restrições de não sobreposição, os autores usaram variáveis binárias que relacionam partições do *no-fit polygon*. Esse modelo foi melhorado por Alvarez-Valdes, Martinez e Tamarit (2013), que mostraram como realizar partições do *no-fit polygon* quando ele é não convexo, além de melhorar os limitantes das variáveis contínuas por meio de uma técnica de *lifting*. Toledo *et al.* (2013) desenvolveram um modelo de programação linear inteira mista que considera o recipiente discretizado sobre uma malha. As variáveis de decisão binárias consideram o ponto de referência dos itens posicionado sobre algum ponto da malha, com os domínios estabelecidos pelo *inner-fit raster*.

Ainda na linha de algoritmos exatos para o 2ISP, em Leão *et al.* (2016), o modelo linear inteiro misto contém variáveis semi-contínuas, com um dos eixos sendo discretizado. Esse modelo considera o *inner-fit polygon* para determinar o domínio das variáveis e o *no-fit polygon* para determinar as restrições de não sobreposição. Cherri *et al.* (2016) propuseram dois modelos, incluindo a possibilidade de rotação e itens com buracos. O primeiro modelo adota

a trigonometria direta para modelar as restrições de não sobreposição entre itens, enquanto o segundo modelo usa o *no-fit polygon*. Os experimentos computacionais mostraram que o segundo modelo apresenta os melhores resultados ao comparar com modelos anteriores da literatura. Rodrigues e Toledo (2017) apresentaram melhorias sobre o modelo de Toledo *et al.* (2013), desenvolvendo restrições de clique para lidar com a não sobreposição entre itens. Essas restrições são obtidas sobre cliques de um grafo de conflitos, em que os vértices representam as variáveis binárias do modelo e as arestas indicam que os vértices se sobrepõem. Recentemente, Leão *et al.* (2020) revisaram os modelos matemáticos, incluindo aqueles resolvidos por algoritmos *branch-and-cut*. Os autores utilizaram um *framework* comum a fim de apontar as diferenças e semelhanças entre cada proposta da literatura. Os autores apontaram que as pesquisas nessa linha ainda são muito limitadas e, assim, existem oportunidades para a melhoria dos modelos existentes, incluindo a proposição de limitantes inferiores apertados, desigualdades válidas, métodos de decomposição e a integração com heurísticas.

Não existem muitos trabalhos na literatura envolvendo problemas de corte e empacotamento com incertezas, em particular, quando os itens são irregulares. Ao melhor do que se conhece, há apenas o trabalho de Mundim (2017), que considerou incertezas na demanda dos itens. Para o problema de corte de estoque, o autor desenvolveu um modelo de programação estocástica de dois estágios. O modelo penaliza a falta e o excesso de itens produzidos a mais que o necessário nos cenários, sendo aplicado sobre seis instâncias inspiradas na indústria têxtil.

3.1.2 Contribuições

Observando a existência de poucos trabalhos que consideram incertezas em problemas de corte e empacotamento de itens irregulares e diante do número de aplicações que o problema de corte em faixa com incertezas na demanda pode possuir, propõe-se um algoritmo *branch-and-cut* que considera a resolução de um modelo de programação estocástica de dois estágios com recurso para o 2ISP-DU. O algoritmo integra uma heurística baseada em busca em vizinhança variável para gerar limitantes iniciais e cortes válidos durante a resolução de nós da árvore de busca. Outros limitantes baseados na área e tamanho dos itens também são utilizados para restringir o tamanho da área da faixa.

Como se adota o *no-fit raster* para gerar restrições de não sobreposição, o modelo de programação estocástica considera a faixa discretizada sobre uma malha de pontos. Posicionar itens em uma malha pode ser vantajoso para a heurística, que pode obter rapidamente muitos planos de corte diferentes e, então, acelerar a convergência do algoritmo *branch-and-cut*. A heurística obtém uma solução a partir de uma sequência de itens posicionada por duas regras baseadas na *bottom-left*. Novas soluções são obtidas a partir de movimentos de troca, inserção e inversão dentro da sequência. Com o objetivo de melhorar a solução, aplica-se uma busca local baseada em movimentos de troca de itens dentro da sequência. Além disso, a heurística é adaptada para completar soluções parciais que advêm dos nós da árvore de busca. Ou seja,

os itens cuja variável de decisão do modelo impõe o posicionamento em determinado ponto da malha são fixados, enquanto a heurística busca posicionar os demais itens cujas variáveis de decisão têm valor fracionário.

O algoritmo *branch-and-cut* é testado em instâncias da literatura com até 65 itens. Para o problema sem incerteza (isto é, o 2ISP), o algoritmo proposto é comparado com outros dois algoritmos, obtendo a solução ótima para cerca de 55% das instâncias dentro de uma hora de execução, enquanto o *gap* médio é de aproximadamente 8%. Por outro lado, para o problema com incertezas na demanda, as instâncias são adaptadas e consideram até 80 cenários. O algoritmo proposto apresenta uma solução ótima para instâncias com mais de 60 cenários em poucas horas. Além disso, na análise do valor esperado da informação perfeita e do valor da solução estocástica, observa-se como as incertezas impactam no problema. Portanto, as incertezas precisam ser tratados adequadamente, uma vez que a solução do problema estocástico apresenta uma vantagem econômica e faz melhor uso da área da faixa.

O restante deste trabalho está organizado da seguinte forma. A seção 3.2 descreve o 2ISP-DU e como o *no-fit raster* é usado. A seção 3.3 apresenta o modelo estocástico desenvolvido para o problema e o método de solução proposto. Experimentos numéricos e seus resultados são discutidos na Seção 3.4, incluindo um estudo do valor esperado da informação perfeita e do valor da solução estocástica. Finalmente, tem-se as considerações finais na Seção 3.5.

3.2 Descrição do problema

O problema de corte de itens irregulares em faixa com demanda incerta (isto é, o 2ISP-DU) é uma extensão do problema de corte de itens irregulares em faixa (2ISP) , que é NP-difícil mesmo quando os itens são retângulos. Esses problemas são definidos sobre o plano Cartesiano. A faixa está posicionada no primeiro quadrante, com origem em $(0,0)$, e tem uma altura (ou comprimento) fixa conhecida H e uma largura desconhecido e ilimitada L. O eixo x está associado à dimensão da largura (L) e o eixo y está associado à dimensão da altura (H). Um item i do conjunto de itens I tem área a_i, um número de cópias (ou demanda) d_i, um vértice de referência e um envelope retangular r_i.

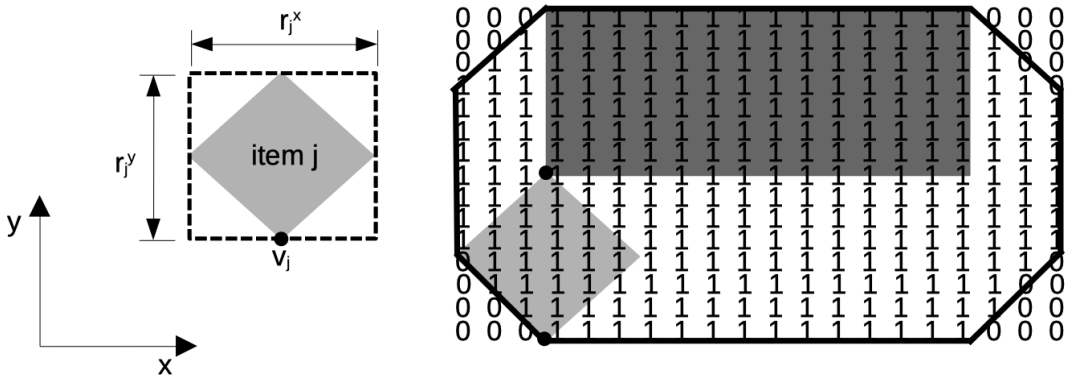
Um item i é posicionado pelo seu vértice de referência $v_i = (v_i^x, v_i^y)$, que corresponde ao vértice de menor ordenada (eixo y) (em caso de empate, o vértice com menor abscissa). A envoltória retangular $r_i = (r_i^x, r_i^y)$ consiste no menor retângulo, em termos de dimensões, sem rotação e paralelo aos eixos, que circunscreve o item i. Os itens são considerados com orientação fixa, isto é, não é permitida a rotação de itens. Na Figura 3.1(a), ilustra-se o vértice de referência do losango pelo ponto preto, enquanto o envelope retangular é o retângulo tracejado que circunscreve o item. O envelope retangular do retângulo é o próprio retângulo. Na Figura 3.1(b), mostram-se as dimensões (L, H) da faixa. Observe que o lado direito da faixa está aberto (linha tracejada), o que significa que a largura L precisa ser determinado ao resolver o problema.

O objetivo do problema é obter um posicionamento viável de todos os itens, atendendo às demandas d_i de cada item i, de forma que a largura total L seja mínima para uma altura H fixa. Uma solução é viável quando todos os itens são posicionados na faixa (ou seja, a demanda dos itens é atendida exatamente); não há sobreposição entre qualquer item (ou seja, os itens são posicionados na faixa de forma a não se sobreporem); e os itens não podem extrapolar as dimensões da faixa (ou seja, nenhuma parte de qualquer item pode ir além dos tamanhos da faixa, exceto pelo lado aberto (direito)). Observe que a largura L (ou seja, o lado aberto (direito)) é determinado pelo vértice mais à direita de um item posicionado na faixa.

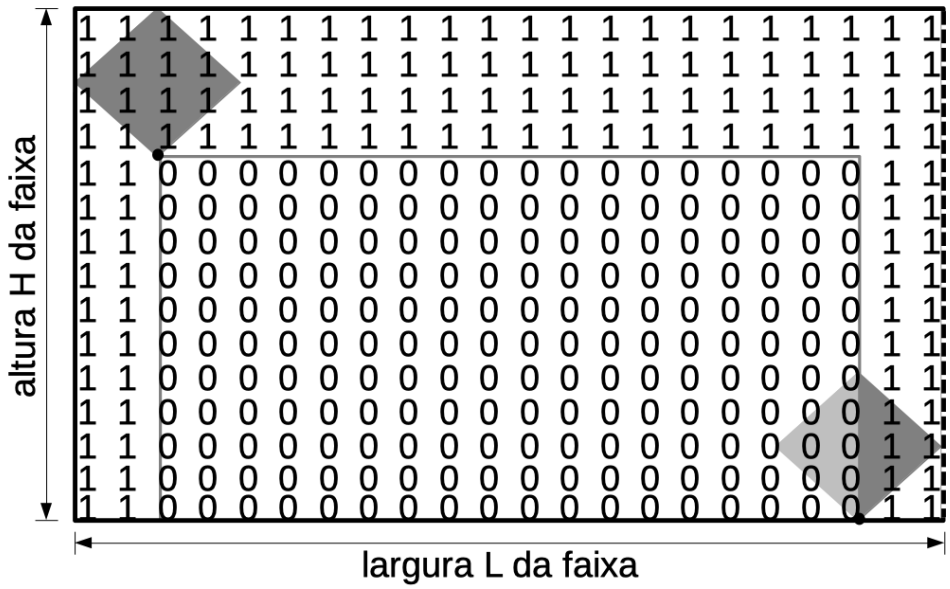
Para garantir que não haja sobreposição entre itens, utliza-se o *no-fit raster* (TOLEDO *et al.*, 2013). O *no-fit polygon* NFP_{ij} é calculado entre cada par de itens i e j, onde i é fixo e j está orbitando em torno de i. A cada NFP_{ij} está associada uma matriz binária cujas células com o valor "um" indicam que i e j estão sobrepostos; caso contrário, tem-se o valor "zero", como se observa na Figura 3.1(a). Para garantir que os itens estão contidos dentro da faixa, emprega-se o *inner-fit raster* (TOLEDO *et al.*, 2013). O *inner-fit polygon* IFP_i é calculado transladando cada item i pelo seu vértice de referência, de modo que i esteja encostando em pelo menos um dos lados da faixa. O *inner-fit raster* é a matriz associada ao IFP_i cujas células com o valor "um" indicam que i estará extrapolando as dimensões da faixa, como se pode ver na Figura 3.1(b). Ao calcular o *inner-fit raster*, usa-se um limite superior para a largura L da faixa.

Considera-se no 2ISP-DU dois momentos para tomar decisões. No primeiro momento (t_0), não se conhece a demanda exata dos itens, apenas uma previsão dessa demanda baseada em dados passados. Nesse momento, pode-se fazer o pedido da área de faixa $H \times L_0$, com custo c_0 por metro quadrado. A demanda dos itens passa a ser conhecida no momento t_1, quando se fecha os pedidos para a posterior produção dos itens, resultando no conjunto de itens I. No momento t_1, pode-se solicitar um acréscimo $H \times L_1$ na área da faixa inicial, com custo $c_1 > c_0$. Por motivos operacionais, toda área adicional, solicitada no momento t_1 custa mais. O objetivo é determinar qual área $H \times L$ de faixa comprar, a um custo mínimo, para produzir o conjunto de itens I. Observe que a altura H da faixa é fixa, conhecida de antemão, então o problema recai em definir a largura L da faixa.

A área da faixa é determinada pela área inicial $H \times L_0$ mais a área adicional $H \times L_1$. Se a demanda dos itens do conjunto I fosse conhecida com certeza no momento t_0, a faixa poderia ser determinada exatamente e teria um custo proporcional a c_0. Embora não se conheça a demanda exata dos itens em I no momento t_0, seria positivo comprar uma área inicial capaz de permitir (parcialmente ou por completo) a produção desses itens, sem ter que comprar muito mais área adicional no momento t_1. O desperdício de faixa não é interessante e, por isso, também não se deseja comprar uma área de faixa excessiva no momento t_0. A Figura 3.2 ilustra uma solução viável para o 2ISP-DU.



(a) NFP_{ij} de um retângulo (item fixo) e um losango (item orbital)



(b) IFP_i de um losango

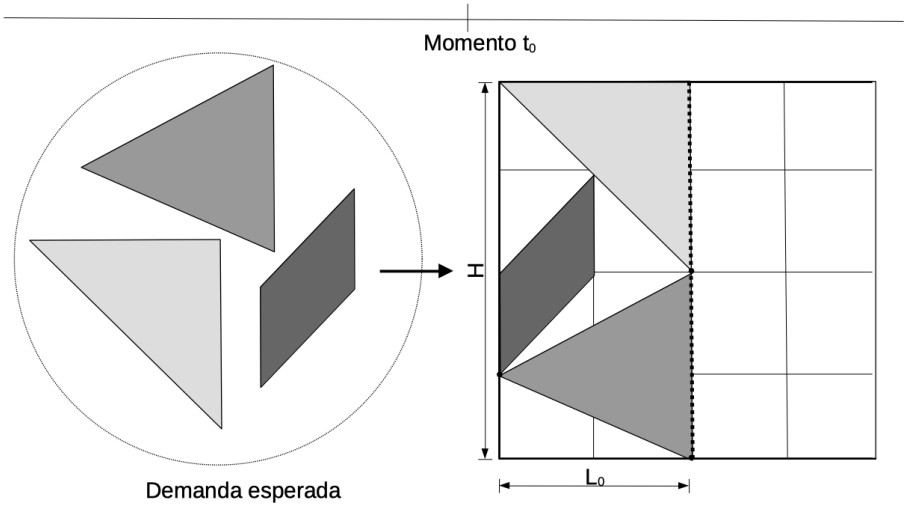
Figura 3.1 – Matrizes de *no-fit raster* e *inner-fit raster*.

3.3 Algoritmo exato

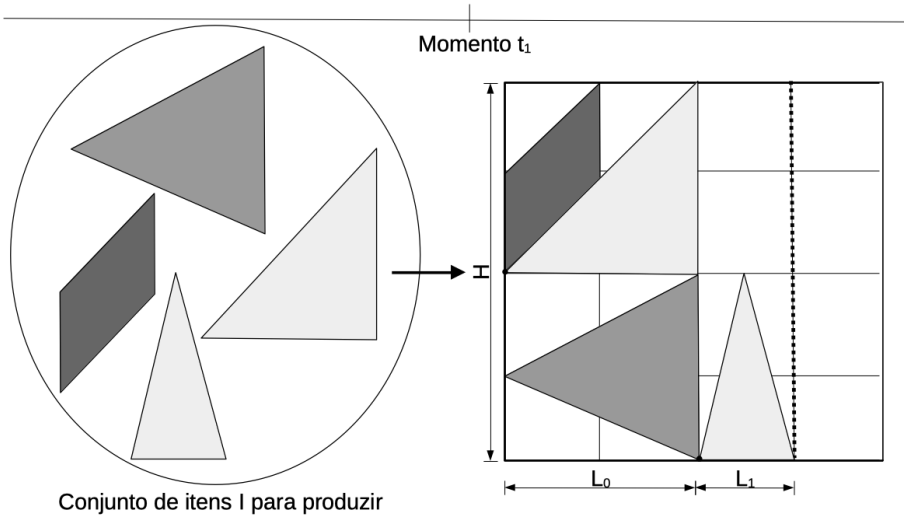
Esta seção traz o algoritmo exato desenvolvido para o 2ISP-DU. Apresenta-se inicialmente o modelo de programação estocástica de dois estágios com recurso, suas variáveis, função objetivo e restrições. Em seguida, discute-se sobre a heurística baseada em busca em vizinhança variável proposta para gerar soluções e limitantes válidos para o problema. Por fim, apresenta-se a estrutura geral do algoritmo *branch-and-cut* e as métricas utilizadas para avaliar a solução do problema estocástico.

3.3.1 Modelo de dois estágios

No 2ISP-DU, a demanda dos itens no conjunto I é considerada uma variável aleatória, com realizações discretas conforme uma distribuição de probabilidade conhecida (BIRGE; LOUVEAUX, 1997). Para tanto, propõe-se para o problema um modelo de programação estocástica



(a) Área da faixa determinada no momento t_0



(b) Área da faixa determinada no momento t_1

Figura 3.2 – Exemplo de uma solução viável para o 2ISP-DU.

de dois estágios com recurso. O conjunto de possíveis estados é $\Omega = \{1, 2, \ldots, S\}$, tal que $s \in \Omega$ define a ocorrência particular de um estado, isto é, um cenário representando os itens e suas demandas em I^s. No primeiro estágio do modelo, determina-se a área inicial da faixa, as decisões "aqui-e-agora" (*here-and-now*). No segundo estágio, após a realização das variáveis aleatórias, têm-se ações para ajustar a área da faixa, isto é, comprar área adicional de faixa (decisões "espere-e-veja" (*wait-and-see*).

O modelo parte da proposta de Toledo *et al.* (2013), que considera o posicionamento dos itens sobre uma malha, fazendo uso do *no-fit raster* e *inner-fit raster* para obter as posições válidas para o posicionamento de itens. Além disso, os parâmetros e as variáveis de decisão do modelo são:

- H: altura da faixa, que é fixa e conhecida;

- c_0: custo unitário de área da faixa, se comprada no instante t_0;
- c_1: custo unitário de área da faixa, se comprada no instante t_1;
- M: número suficientemente grande;
- π^s: probabilidade de ocorrência do cenário $s \in \Omega$;
- I^s: conjunto com a demanda dos itens irregulares do cenário $s \in \Omega$.
- L_0: variável contínua que denota a largura inicial de faixa a ser definida no momento t_0;
- x^s_{ipq}: variável binária que tem valor 1 se o item $i \in I^s$, do cenário $s \in \Omega$, tem o seu vértice de referência posicionado no ponto (p,q) da malha associada a faixa. Caso contrário, ela recebe o valor 0;
- L^s_1: variável contínua que denota a largura adicional de faixa para suprir a demanda dos itens em I^s, dado o cenário $s \in \Omega$.

O 2ISP-DU é formulado como um modelo de programação estocástica de dois estágios e risco neutro. O objetivo é minimizar o custo total associado à compra da faixa, considerando a produção da demanda esperada para os itens em I.

O modelo de primeiro estágio tem função objetivo (3.1), que está relacionada com o custo da área da faixa adquirida no momento t_0, e os custos esperados pela aquisição de área da faixa no momento t_1, dado o problema de segundo estágio com os cenários $s \in \Omega$. O vetor $\boldsymbol{\xi} = [\xi_s]$ contém os parâmetros estocásticos do modelo, em que $\xi_s = \{L^s_1, x^s_{ipq}\}$ representa uma realização desse vetor através do cenário s. A função de recurso $Q(L_0, \boldsymbol{\xi})$ fornece o custo total esperado de área da faixa adicional pela resolução do problema de segundo estágio. Esse custo esperado é aproximado pelo conjunto amostrado de cenários em Ω. O domínio das variáveis de decisão de primeiro estágio é dado em (3.2).

$$\text{Minimizar } c_0(HL_0) + Q(L_0, \boldsymbol{\xi}) \tag{3.1}$$

$$\text{sujeito a: } L_0 \geq 0. \tag{3.2}$$

Por outro lado, o modelo de segundo estágio possui função objetivo (3.3), que representa a minimização da soma dos custos esperados de área adicional de faixa, dada a probabilidade π^s

de ocorrência do cenário $s \in \Omega$.

$$Q(L_0, \boldsymbol{\xi}) = \text{Minimizar} \sum_{s \in \Omega} \pi^s (c_1(HL_1^s)) \tag{3.3}$$

$$\text{sujeito a: } (p + (r_i^x - v_i^x))x_{ipq}^s \leq L_0 + L_1^s, \qquad \forall\, s \in \Omega,\ i \in I^s,\ (p,q) \in IFP_i; \tag{3.4}$$

$$\sum_{j \in I^s} \sum_{(u,v) \in NFP_{ij}^{(p,q)}} x_{juv}^s \leq (1 - x_{ipq}^s)M, \qquad \forall\, s \in \Omega,\ i \in I^s,\ (p,q) \in IFP_i; \tag{3.5}$$

$$\sum_{(p,q) \in IFP_i} x_{ipq}^s = d_i^s, \qquad \forall\, s \in \Omega,\ i \in I^s; \tag{3.6}$$

$$L_1^s \geq 0, \qquad \forall\, s \in \Omega; \tag{3.7}$$

$$x_{ipq}^s \in \{0,1\}, \qquad \forall\, s \in \Omega,\ i \in I^s;\ (p,q) \in IFP_i. \tag{3.8}$$

No modelo de segundo estágio, as restrições (3.4) asseguram que a largura adicional L_1^s do cenário $s \in \Omega$ é determinada pelo item i, dentre todos aqueles de I^s, que tem o seu vértice de referência posicionado em (p,q) e seu término em $p + (r_i^x - v_i^x)$, levando em consideração a largura inicial L_0 da faixa. As restrições (3.5) asseguram que se o item $i \in I^s$, para cada cenário $s \in \Omega$, está posicionado em (p,q), então outro item j não pode ficar posicionado em qualquer um dos pontos (u,v) do conjunto $NFP_{ij}^{(p,q)}$. As restrições (3.6) impõem que a demanda de cada item $i \in I^s$, para cada cenário $s \in \Omega$, seja atendida exatamente. Por fim, para cada cenário $s \in \Omega$, as restrições (3.7) e (3.8) definem que a largura adicional L_1^s é não-negativa e as variáveis x_{ipq}^s são binárias, respectivamente. Pode-se considerar ainda restrições adicionais para ajudar na não sobreposição de itens, impondo que cada ponto da malha seja coberto por no máximo um item (JUNQUEIRA, 2009).

Nas restrições (3.5) e (3.6), define-se o $NFP_{i,j}^{(p,q)}$ como sendo o conjunto de pontos (u,v) que estão no interior do *no-fit raster* entre i (item fixo) e j (item orbital), dado que o item i está posicionado em (p,q). Os pontos (u,v) são aqueles em que o item j, se posicionado em algum deles, faz com que haja sobreposição com i posicionado em (p,q). No modelo de programação estocástica, o valor ótimo corresponde a uma composição de custos mínimos, conforme os cenários e as suas probabilidades de ocorrência, resultando em um custo total mínimo esperado. Assim, resolve-se por um algoritmo *branch-and-cut* o modelo equivalente determinístico (do modelo de programação estocástico), que é dado pela função objetivo (3.1), em que $Q(L_0, \boldsymbol{\xi})$ é a função (3.3), e tem restrições (3.2) e (3.4)-(3.8).

Para avaliar a melhoria que esse modelo estocástico produz sobre um modelo puramente determinístico, pode-se calcular o Valor Esperado da Informação Perfeita (*Expected Value of Perfect Information - EVPI*) e o Valor da Solução Estocástica (*Value of Stochastic Solution - VSS*) (BIRGE; LOUVEAUX, 1997). O EVPI representa o quanto o decisor estaria disposto a pagar em troca de informação perfeita e precisa sobre o futuro. Com essa medida, pode-se verificar a importância de considerar (ou não) a aleatoriedade. Assim, se o valor do EVPI é baixo, pode-se não ser tão importante considerar as incertezas no problema. Para tanto, leva-se em consideração o Valor do Problema Estocástico (*Recourse Problem - RP*) , que é a solução

ótima do modelo equivalente determinístico, e o valor esperado das soluções *wait-and-see - WS* , sendo $\text{WS} = \sum_{s\in\Omega} \pi_s \text{WS}^*_s$, tal que WS^*_s é o valor ótimo do problema *wait-and-see* para cada cenário $s \in \Omega$. Ou seja, resolve-se o modelo equivalente determinístico considerando apenas o cenário s. Logo, $\text{EVPI} = \text{RP} - \text{WS}$.

O VSS representa o custo de ignorar a aleatoriedade dos parâmetros na escolha de uma decisão. Assim, se o valor de VSS é baixo, o ganho ao se resolver o problema estocástico RP não é interessante em comparação a um problema de Valor Esperado (*Expected Value problem - EV*) . Ou seja, quão ruim é a decisão EV em comparação com a solução de RP. O EV representa um problema em que as variáveis aleatórias são substituídas pelos seus respectivos valores esperados (que pode ser de um cenário de referência). A partir da solução do EV, obtém-se o resultado esperado, isto é, a *Expectation of the Expected Value Problem - EVV* . O valor EVV é obtido ao fixar as variáveis de primeiro estágio pela solução dada pelo EV e, assim, resolver o problema estocástico RP. Logo, $\text{VSS} = \text{EVV} - \text{RP}$.

3.3.2 Soluções viáveis: busca em vizinhança variável

Com o objetivo de gerar soluções e limitantes válidos para o algoritmo *branch-and-cut*, desenvolve-se uma busca em vizinhança variável (*Variable Neighborhood Search* - VNS) para o problema. Esse método é atraente em comparação a outras heurísticas porque permite diferentes formas de codificar e decodificar uma solução devido às suas estruturas internas e a facilidade para inserir novos componentes. A VNS desenvolve a otimização sobre uma única solução, com uma mudança sistemática de vizinhanças à medida que a solução estagna em um ótimo local (HANSEN; MLADENOVIĆ; PÉREZ, 2010).

No caso particular de problemas de corte de itens irregulares, Souza Queiroz e Andretta (2020b) desenvolveram uma versão geral da VNS para o problema da mochila, concluindo que a etapa de busca local pode demandar bastante tempo computacional. Diante disso, desenvolve-se uma versão básica da VNS, com a solução do problema sendo representada por um vetor de números inteiros. As estruturas de vizinhança são baseadas em movimentos de troca, inserção e inversão de posições dentro desse vetor, diferentemente de Souza Queiroz e Andretta (2020b) que consideraram apenas movimentos de troca e inserção. O posicionamento dos itens segue uma lista de pontos gerada por duas regras baseadas na *bottom-left*. O Algoritmo 3.1 ilustra a estrutura da VNS implementada.

No Algoritmo 3.1, a solução é representada por um vetor s com n inteiros. Cada posição guarda o índice de um item do conjunto de itens disponíveis para cortar. A solução inicial é gerada aleatoriamente atribuindo o índice de cada item a uma posição do vetor s. O posicionamento dos itens segue a sequência (da esquerda para a direita) em que eles se encontram no vetor solução. No laço das linhas 5-12, gera-se inicialmente uma solução s', de forma aleatória, na vizinhança corrente $N_k(s)$. Em seguida, a solução s' passa por uma busca local que compreende movimentos de troca dentro do vetor solução, objetivando a melhor solução que possa ser obtida

Algoritmo 3.1: Busca em Vizinhança Variável Básica.

```
1  início
2      Gerar aleatoriamente uma solução inicial s.
3      enquanto critério de parada não for atingido faça
4          k ← 1.
5          while k ≤ K_max do
               /*gerar uma nova solução
6              s' ← gerar uma solução aleatoriamente na vizinhança N_k(s).
               /*busca local - tentar todos os movimentos de troca
7              s'' ← gerar a melhor solução na vizinhança N_1(s).
8              se f(s'') > f(s) então
9                  s ← s''.
10                 k ← 1.
11             senão
12                 k ← k + 1.
13     retorna s.
```

realizando esses movimentos. Sempre que um movimento de troca melhora a solução, reinicia-se a execução de todos os movimentos de troca sobre a nova solução, repetindo isso até que nenhum movimento de troca possa melhorar a solução. No caso da solução resultante da busca local s'' ser melhor do que a solução corrente s, atualiza-se a solução corrente e a busca reinicia na primeira vizinhança; caso contrário, a busca prossegue para a próxima vizinhança.

O valor de um vetor solução s é dado pela função $f(s)$, sendo preciso transformar o vetor s em uma solução do problema. Para tanto, utiliza-se o decodificador descrito no Algoritmo 3.2. Com relação as estruturas de vizinhança, propõe-se utilizar três, definidas a partir de testes numéricos preliminares: N_1 representa um movimento de troca, sendo dois elementos do vetor solução escolhidos (aleatoriamente) e trocados de posição; N_2 representa um movimento de inserção, sendo dois elementos do vetor solução escolhidos (aleatoriamente), tal que o primeiro elemento é inserido em posição imediatamente anterior ao segundo elemento; N_3 representa um movimento de inversão, sendo duas posições do vetor solução escolhidas (aleatoriamente), tal que a subsequência entre essas duas posições é considerada em ordem inversa. A busca local que ocorre na linha 7 do Algoritmo 3.1 corresponde a estrutura de vizinhança N_1, sendo testada todas as possibilidades de troca entre dois elementos do vetor solução.

No Algoritmo 3.2, considera-se a melhor solução em B_{sol}, que inicialmente tem valor infinito, a tabela *hash* T, que está inicialmente vazia, e um vetor s, que representa a solução atual. Inicialmente, verifica-se se esse vetor já foi analisado e se encontra-se na tabela *hash* T. Se há uma solução associada ao vetor s, então ela é retornada e corresponde a área total da faixa. Por outro lado, o laço das linhas 5-12 considera o posicionamento dos itens por cada regra de posicionamento r, resultando na lista de pontos LP. No laço interno das linhas 7-10, para cada item $s[i]$ do vetor solução, busca-se na lista LP o primeiro ponto (a,b) onde o vértice de

Algoritmo 3.2: Decodificador de um vetor solução s com n itens.

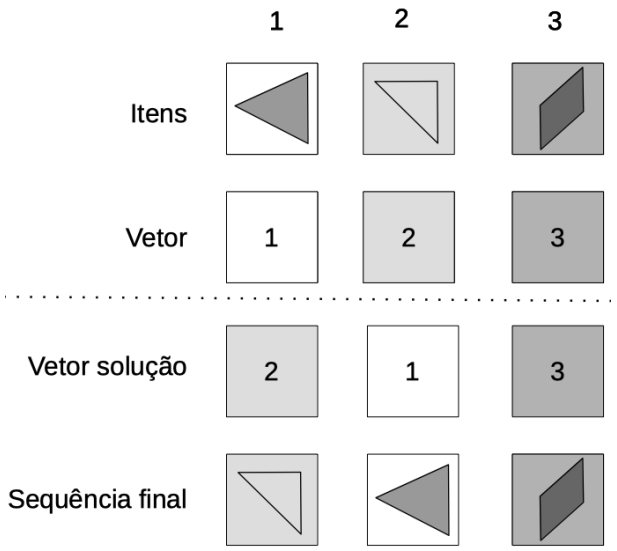
```
/*Sol_r é a solução gerada com a r-ésima regra de posicionamento; f()
retorna o valor de uma solução; T é a tabela hash com os vetores
solução já avaliados.
1  início
2      B_sol ← ∅.
3      se s está salva na tabela T então
4          retorna valor da solução s salva em T.
5      para r ← 1,2 faça
6          LP ← aplicar a regra de posicionamento r e obter a lista de pontos para
             posicionar os itens.
7          para i ← 1,2,...,n faça
8              (a,b) ← ponto de LP que resulte na menor área de faixa ocupada ao
                 posicionar o item s[i].
9              Sol_r ← Sol_r ∪ {s[i],(a,b)}.
10             Remover de LP os pontos sobrepostos pelo item s[i] posicionado em (a,b).
11         se f(Sol_r) < f(B_sol) então
12             B_sol ← Sol_r.
13     Salvar a solução B_sol na tabela T.
14     retorna B_sol.
```

referência de $s[i]$ resultará na menor área ocupada de faixa. Após posicionar o item nesse ponto, removem-se os pontos de LP que serão sobrepostos por ele. Após posicionar todos os itens, a solução Sol_r tem seu valor (isto é, a área ocupada da faixa) comparado com a melhor solução B_{sol}. Ao final, a solução em B_{sol} é salva na tabela T.

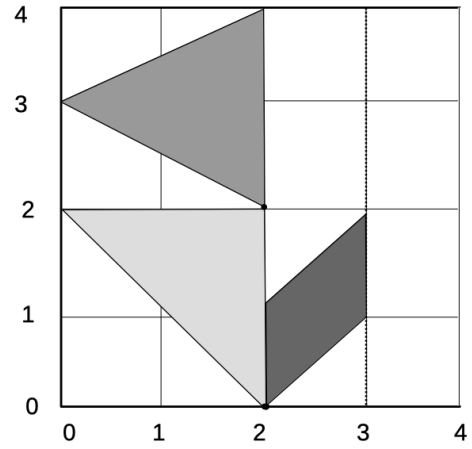
Duas regras de posicionamento são utilizadas pelo Algoritmo 3.2 para obter a lista de pontos LP. A primeira regra é a *bottom-left*, em que os pontos são adicionados a LP de baixo para cima e da esquerda para a direita. A outra regra é a *top-left* em que os pontos são adicionados a LP de cima para baixo e da esquerda para a direita. Isso significa que cada item é posicionado na faixa da esquerda para a direita, seguindo cada lista de pontos (isto é, de baixo para cima ou de cima para baixo), no primeiro ponto que resultar na menor área ocupada de faixa com o dado item. Os testes preliminares apontaram melhores soluções quando essas duas regras são utilizadas, ao invés de apenas uma delas.

A Figura 3.3 ilustra as soluções geradas ao utilizar cada regra de posicionamento. Na Figura 3.3a há um exemplo de vetor com três itens (1, 2 e 3) cuja sequência final de posicionamento é 2, 1 e 3. A solução pela regra *bottom-left* é ilustrada na Figura 3.3b, seguindo a lista de pontos LP para posicionar os itens. O item 2 é posicionado no primeiro ponto viável que gera a menor largura de faixa, isto é, o ponto $(2,0)$, assim eliminando os pontos de LP que esse item cobre. Em seguida, para o item 1, tem-se o ponto $(2,2)$, eliminando de LP os pontos cobertos por esse item. Por fim, o primeiro ponto viável que gera a menor largura de faixa para o item

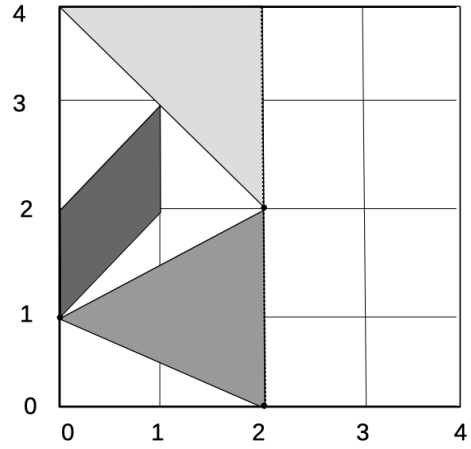
3 é o $(2,2)$, resultando numa faixa de largura total 3. Por outro lado, na Figura 3.3c, tem-se a solução obtida com a regra *top-left*. Nota-se que foi possível posicionar o item 3 no espaço vazio entre os itens 1 e 2, gerando uma faixa cuja largura total é igual a 2.



(a) Vetor solução



(b) Solução pela regra *bottom-left*



(c) Solução pela regra *top-left*

Figura 3.3 – Decodificação de um vetor solução pela VNS.

3.3.3 Algoritmo branch-and-cut

O algoritmo exato proposto para o 2ISP-DU considera o *framework* de *branch-and-cut* do Gurobi (GUROBI OPTIMIZATION, 2020). Inicialmente, para cada cenário $s \in \Omega$, aplica-se a VNS para obter uma solução inicial Sol_s, com largura L^s_{ub} da faixa. Dessa forma, essa solução e largura são utilizadas para fornecer uma solução inicial e limitantes para as variáveis do modelo equivalente determinístico, para cada $s \in \Omega$. Ainda considerando cada cenário s, calcula-se a menor largura de faixa L_f da seguinte forma: (i) dentre os itens em I^s, faça l^s receber a largura

do item de maior largura (isto é, $\max_{i \in I^s}(r_i^x)$); e (ii) faça a^s receber a soma da área dos itens em I^s dividida pela altura H da faixa (isto é, $\sum_{i \in I^s}(a_i)/H$). Logo, L_{lb}^s recebe o maior valor entre l^s e a^s, resultando no limitante inferior inicial $L_0 + L_1^s \geq L_{lb}^s$ para o cenário s, onde L_0 e L_1^s são as variáveis do modelo.

No *framework* de *branch-and-cut* do Gurobi, considera-se a possibilidade de usar todas as *threads* do processador, a inserção de cortes (pela função *addLazy*) e a inserção das soluções da VNS (pela função *setSolution*). Nesse último caso, uma *callback* é chamada conforme o número de nós avaliados da árvore de busca. Nos experimentos preliminares, observou-se que ao chamar constantemente a VNS, tem-se um aumento no tempo de resolução, a geração de cortes dominados e pouca ação do *solver* na exploração de nós da árvore. Por isso, considera-se que a *callback* é chamada nas seguintes condições: até os 1000 nós, aplica-se a VNS a cada 250 nós explorados; entre 1000 e 10000 nós, aplica-se a VNS a cada 1000 nós explorados; entre 10000 e 100000 nós, aplica-se a VNS a cada 5000 nós explorados; caso contrário, aplica-se a VNS a cada 10000 nós explorados.

Para esses nós onde a *callback* atua, dado o vetor $\bar{x}$ com a solução ótima da relaxação linear, para cada cenário s, obtém-se a solução parcial $\overline{Sol}^s$ com os itens i e os pontos (p,q) se $\bar{x}_{ipq}^s \geq 0,99$. Em seguida, essa solução é passada para a VNS de forma que esses itens de $\overline{Sol}^s$ são fixados na faixa e não fazem parte do vetor solução. Ou seja, a VNS considerará o vetor solução formado apenas com os itens de I^s que não estão em $\overline{Sol}^s$. Com isso, no momento de decodificação do vetor solução, posiciona-se primeiramente os itens em $\overline{Sol}^s$ para, em seguida, posicionar os itens do vetor solução conforme o Algoritmo 3.2. Ao final, para cada cenário s, a solução gerada pela VNS é adicionada no *solver* e a largura dessa solução L_{no}^s resulta na desigualdade $L_0 + L_1^s \leq L_{no}^s$ válida para o problema.

3.4 Experimentos computacionais

O objetivo dos experimentos numéricos é avaliar a performance do algoritmo *branch-and-cut* bem como o impacto em se considerar incertezas no problema. Todos os algoritmos foram codificados na linguagem C++, juntamente com os métodos para calcular o *inner-fit raster* e *no-fit raster*. O *framework* do algoritmo *branch-and-cut* considera a versão 9.1 do Gurobi, com suas configurações padrões, exceto as mencionadas na Seção 3.3.3. O computador usado nos experimentos tem sistema operacional Linux Ubuntu 16.04 LTS, processador Intel Xeon X5660 de 2,8 GHz e 48 GB de RAM.

3.4.1 Calibração e instâncias

Pelo fato da heurística VNS fazer parte do algoritmo exato e ser chamada durante a otimização como procedimento auxiliar, optou-se por realizar a calibragem dos parâmetros via tentativa e erro, objetivando balancear a qualidade da solução com o tempo computacional

exigido neste processo. Dessa forma, adota-se três critérios de parada: um número máximo de iterações MAX, um número máximo de iterações consecutivas sem melhora da solução $CONS$ e um tempo limite $TIME$. O primeiro critério que for alcançado finaliza a heurística. Para gerar a solução inicial e os limitantes iniciais válidos para o algoritmo *branch-and-cut*, esses parâmetros foram definidos como: $MAX = 250$, $CONS = 25$ e $TIME = 120$ segundos. Por outro lado, quando a heurística é chamada para gerar soluções válidas nos nós da árvore de busca, esses parâmetros tiveram os seguintes valores: $MAX = 25$, $CONS = 3$ e $TIME = 5$ segundos. Outros parâmetros adotados no algoritmo exato foram descritos na Seção 3.3.3.

Os experimentos computacionais consideram 44 instâncias da literatura. Algumas dessas instâncias podem ser encontradas na página do *EURO Special Interest Group on Cutting and Packing - ESICUP*. Alguns dados das instâncias são informados na Tabela 3.1, como os autores que as propuseram, o número total de itens, a altura da faixa e o limitante inferior válido para a largura da faixa conforme calculado na Seção 3.3.3. A escala adotada para obter as matrizes de *inner-fit raster* e *no-fit raster* é de 1 ponto para cada 1 unidade de distância. Essas instâncias estão de acordo com a proposta de algoritmos exatos para o 2ISP (LEÃO *et al.*, 2020).

Tabela 3.1 – Dados das instâncias utilizadas nos experimentos.

Instância	Autor	#Itens	H	L_{lb}	Instância	Autor	#Itens	H	L_{lb}
blasz2	Oliveira, Gomes e Ferreira (2000)	16	15	16	poly1e	Rodrigues e Toledo (2017)	15	40	10
blazewicz1	Toledo *et al.* (2013)	7	15	6	rco1	Toledo *et al.* (2013)	7	15	7
blazewicz2	Toledo *et al.* (2013)	14	15	11	rco2	Toledo *et al.* (2013)	14	15	13
blazewicz3	Toledo *et al.* (2013)	21	15	17	rco3	Toledo *et al.* (2013)	21	15	19
blazewicz4	Toledo *et al.* (2013)	28	15	22	rco4	Toledo *et al.* (2013)	28	15	26
blazewicz5	Toledo *et al.* (2013)	35	15	27	rco5	Toledo *et al.* (2013)	35	15	32
dagli1	Rodrigues e Toledo (2017)	10	60	23	shapes2	Toledo *et al.* (2013)	8	40	14
dighe1	Dighe e Jakiela (1995)	16	100	100	shapes4	Toledo *et al.* (2013)	16	40	16
dighe2	Dighe e Jakiela (1995)	10	100	100	shapes5	Toledo *et al.* (2013)	20	40	20
fu	Fujita, Akagi e Hirokawa (1993)	12	38	29	shapes7	Toledo *et al.* (2013)	28	40	28
fu5	Alvarez-Valdes, Martinez e Tamarit (2013)	5	38	14	shapes9	Toledo *et al.* (2013)	34	40	33
fu6	Alvarez-Valdes, Martinez e Tamarit (2013)	6	38	17	shapes15	Toledo *et al.* (2013)	43	40	40
fu7	Alvarez-Valdes, Martinez e Tamarit (2013)	7	38	19	shirts1-2	Rodrigues e Toledo (2017)	13	40	13
fu8	Alvarez-Valdes, Martinez e Tamarit (2013)	8	38	20	shirts2-4	Rodrigues e Toledo (2017)	26	40	14
fu9	Alvarez-Valdes, Martinez e Tamarit (2013)	9	38	23	shirts3-6	Rodrigues e Toledo (2017)	39	40	21
fu10	Alvarez-Valdes, Martinez e Tamarit (2013)	10	38	26	shirts4-8	Rodrigues e Toledo (2017)	52	40	28
han	Han e Na (1996)	23	58	34	shirts5-10	Rodrigues e Toledo (2017)	65	40	35
jakobs1	Jakobs (1996)	25	40	10	three	Alvarez-Valdes, Martinez e Tamarit (2013)	3	7	4
poly1a	Hopper (2000)	15	40	13	threep2	Alvarez-Valdes, Martinez e Tamarit (2013)	6	7	7
poly1b	Rodrigues e Toledo (2017)	15	40	13	threep2w9	Alvarez-Valdes, Martinez e Tamarit (2013)	6	9	6
poly1c	Rodrigues e Toledo (2017)	15	40	13	threep3	Alvarez-Valdes, Martinez e Tamarit (2013)	9	7	10
poly1d	Rodrigues e Toledo (2017)	15	40	11	threep3w9	Alvarez-Valdes, Martinez e Tamarit (2013)	9	9	8

Por não serem conhecidas instâncias para o 2ISP-DU, utilizaram-se os dados das instâncias na Tabela 3.1. Considera-se o custo inicial de área da faixa $c_0 = 1$ por metro quadrado e o custo adicional de área de faixa $c_1 = 1,5$ por metro quadrado (isto é, 50% a mais que o custo inicial). Esses valores foram definidos após experimentos preliminares, notando que valores mais altos (menores) de c_1 em comparação com c_0 tendiam a soluções tendo correspondentemente uma área de faixa inicial maior (respectivamente, menor). A geração dos cenários é feita por duas abordagens: a construção de uma árvore de cenários (MA; KWON; LEE, 2010) e a enumeração de realizações discretas e equiprováveis de cenários (PAN; NAGI, 2010). Em ambos os casos, separam-se os n itens da instância em dois grupos. O Grupo 1 contém os primeiros $\lfloor n/2 \rfloor$ itens, enquanto o Grupo 2 possui os demais itens, incluindo as suas cópias. Essa partição em grupos é necessária para definir um tipo de demanda para cada item e assim caracterizar o cenário.

Tabela 3.2 – Árvore de cenários para os grupos de itens.

Grupo 1	Grupo 2	Cenários	Probabilidade			
			Moderado	Equiprovável	Otimista	Pessimista
Baixa	Baixa	Baixa-Baixa	6,25%	11,09%	36,00%	1,00%
	Média	Baixa-Média	12,50%	11,12%	18,00%	3,00%
	Alta	Baixa-Alta	6,25%	11,09%	6,00%	6,00%
Média	Baixa	Média-Baixa	12,50%	11,12%	18,00%	3,00%
	Média	Média-Média	25,00%	11,16%	9,00%	9,00%
	Alta	Média-Alta	12,50%	11,12%	3,00%	18,00%
Alta	Baixa	Alta-Baixa	6,25%	11,09%	6,00%	6,00%
	Média	Alta-Média	12,50%	11,12%	3,00%	18,00%
	Alta	Alta-Alta	6,25%	11,09%	1,00%	36,00%

Consideram-se três tipos de demandas; ou seja, cada grupo possui itens com o tipo Baixa, Média ou Alta de demanda. Portanto, para um determinado cenário, a demanda de um item i é obtida a partir de uma distribuição uniforme discreta em $[0,0;\ 0,4]d_i$ (se o tipo é Baixa), $[0,8;\ 1,2]d_i$ (se o tipo é Médio) e $[1,6;\ 2,0]d_i$ (se o tipo é Alto), sendo d_i a demanda média conhecida do item. É importante mencionar que a demanda em um cenário foi gerada independentemente da demanda em outros cenários.

A Tabela 3.2 apresenta a árvore de cenários a partir da combinação das três realizações de demandas (Baixa, Média e Alta) para cada grupo de itens, totalizando 9 cenários a serem criados por instância. Baseando-se em Alem e Morabito (2015a), cada instância está associada a quatro casos, cada um definindo a probabilidade de ocorrência de cada um dos 9 cenários, ou seja, moderado, em que cenários com o tipo de demanda Média apresentam maior probabilidade de ocorrência; equiprovável, onde todos os cenários têm aproximadamente a mesma probabilidade de ocorrer; otimista, onde cenários com a demanda do tipo Baixo têm maior probabilidade de ocorrer; e pessimista, em que cenários com a demanda do tipo Alto apresentam maior probabilidade de ocorrência. Em cada um desses casos, as probabilidades de ocorrência dos cenários estão relacionadas com o tipo de realização das demandas em cada grupo. No caso moderado, assume-se que a realização das demandas do tipo Baixa, Média e Alta fornecem as probabilidades de 25%, 50% e 25%, respectivamente, para calcular a probabilidade do cenário. Logo, a probabilidade de ocorrência de cada cenário é obtida pela regra do produto. Por exemplo, no caso moderado, o cenário com os tipos de demandas Baixa para o Grupo 1 e Baixa para o Grupo 2 (isto é, Baixa-Baixa), tem probabilidade $25\% \times 25\% = 6,25\%$ de ocorrência. Para os demais casos, associada a realização das demandas do tipo Baixa, Média e Alta, respectivamente, têm-se as probabilidades: 33%, 34% e 33% no caso equiprovável; 60%, 30% e 10% no caso otimista; e, 10%, 30% e 60% no caso pessimista.

A segunda abordagem usada para gerar os cenários consiste na enumeração discreta e equiprovável (isto é, todos os cenários têm a mesma probabilidade de ocorrência). Nesse caso, consideram-se instâncias com 20, 40, 60 e 80 cenários que foram gerados conforme a

coluna "Cenários" da Tabela 3.2 para os Grupos 1 e 2 de itens. Ou seja, os cenários são gerados sequencialmente do primeiro (Baixa-Baixa), passando por cada um dos 9 cenários, até completar o total desejado para a instância. Por exemplo, uma instância com 20 cenários contém cada um dos 9 cenários duas vezes, exceto que cenários do tipo Baixa-Baixa e Baixa-Média vão aparecer três vezes.

3.4.2 Problema determinístico

Embora o objetivo desse trabalho não seja resolver o 2ISP, que é o problema sem incertezas, fez-se uma comparação do algoritmo *branch-and-cut* com dois métodos exatos publicados na literatura. A comparação é feita com os algoritmos *branch-and-cut* de Cherri *et al.* (2016) e de Rodrigues e Toledo (2017), com os melhores resultados conhecidos em termos de métodos exatos para o 2ISP. É importante ressaltar que o modelo de Cherri *et al.* (2016) adota variáveis contínuas para o posicionamento dos itens, enquanto o modelo de Rodrigues e Toledo (2017) trabalha sobre uma malha de pontos, adotada igual a do algoritmo *branch-and-cut* proposto. Os resultados da comparação são apresentados na Tabela 3.3, com o nome da instância, a solução encontrada pelo algoritmo, o *gap* retornado pelo *solver* e o tempo de resolução. É importante destacar que todos os algoritmos limitaram o tempo de otimização em 3600 segundos (1 hora) por instância, porém cada autor utilizou um computador diferente em seus experimentos inviabilizando uma comparação direta no tempo de resolução. As melhores soluções estão marcadas em negrito.

Cherri *et al.* (2016) e Rodrigues e Toledo (2017) não consideraram todas as instâncias da Tabela 3.3 em seus experimentos computacionais. Por isso, a comparação entre os algoritmos leva em consideração as mesmas instâncias resolvidas por cada um deles. Os resultados do algoritmo *branch-and-cut* proposto são competitivos em comparação com a literatura. Para as mesmas 15 instâncias de Cherri *et al.* (2016), em média, o algoritmo proposto retorna o *gap* e o tempo de computação de 3,88% e 1088,70 segundos, com o desvio padrão de 8,21 % e 1590,56 segundos. O algoritmo proposto obtém uma solução ótima para 12 instâncias, embora haja uma diferença no valor ótimo em comparação com Cherri *et al.* (2016) nas instâncias fu5, fu10, threep2 e threep3w9. Como esses autores não consideraram os itens posicionados sobre uma malha, suas soluções (ou seja, o tamanho da faixa) podem ser menores em alguns casos.

Rodrigues e Toledo (2017) consideraram 41 instâncias da Tabela 3.3. Para tais instâncias, o algoritmo proposto obtém 25 soluções ótimas e uma solução viável para todas as outras. Por outro lado, Rodrigues e Toledo (2017) não apresentaram qualquer solução viável para duas instâncias (shapes15 e shirts5-10), embora tenham retornado uma solução ótima para 25 instâncias. Para as soluções não ótimas, o algoritmo proposto retorna uma largura de faixa menor para as instâncias fu, shapes7, shapes9, shapes15 e shirts5-10. Em média, esses autores apresentaram *gap* de 9,03%, com desvio padrão de 22,19%. Por outro lado, o algoritmo proposto retorna, em média, um *gap* de 7,69% e o tempo de 1804,32 segundos, com um desvio padrão de

Tabela 3.3 – Comparação com a literatura do 2ISP em métodos exatos.

	Algoritmo *Branch-and-Cut*			Cherri *et al.* (2016)			Rodrigues e Toledo (2017)		
Instância	L_{ub}	*gap* (%)	Tempo (s)	L_{ub}	*gap* (%)	Tempo (s)	L_{ub}	*gap* (%)	Tempo (s)
blasz2	**26**	0,00	451,30				**26**	0,00	19,62
blazewicz1	**8**	0,00	7,43				**8**	0,00	0,01
blazewicz2	**14**	0,00	195,39				**14**	0,00	4,17
blazewicz3	21	4,76	3600,00				**20**	0,00	1139,96
blazewicz4	28	17,86	3600,00				**27**	3,70	3600,00
blazewicz5	35	22,86	3600,00				**34**	5,88	3600,00
dagli1	**23**	0,00	1,48				**23**	0,00	100,73
dighe1	125	20,00	3600,00	**122,75**	18,54	3600,00			
dighe2	**100**	0,00	134,16	**100**	0,00	37,70			
fu	34	14,71	3600,00	**33,14**	14,00	3600,00	37	16,22	3600,00
fu5	18	0,00	10,87	**17,89**	0,00	0,01	18	0,00	0,11
fu6	**23**	0,00	77,35	**23**	0,00	0,01	**23**	0,00	2,11
fu7	**24**	0,00	196,23	**24**	0,00	0,10	**24**	0,00	1,30
fu8	**24**	0,00	239,73	**24**	0,00	0,10	**24**	0,00	6,90
fu9	**25**	0,00	1345,03	**25**	0,00	5,90	**25**	0,00	129,74
fu10	29	0,00	3519,68	**28,68**	0,00	278,60	29	0,00	1032,96
han	**44**	22,73	3600,00						
jakobs1	12	16,67	3600,00				**11**	0,00	181,30
poly1a	17	23,53	3600,00	**16,35**	20,49	3600,00	17	17,65	3600,00
poly1b	**20**	20,00	3600,00				**20**	20,00	3600,00
poly1c	**13**	0,00	63,78				**13**	0,00	152,25
poly1d	**13**	15,38	3600,00				**13**	15,38	3600,00
poly1e	**12**	16,67	3600,00				**12**	8,33	3600,00
rco1	**8**	0,00	2,07				**8**	0,00	0,01
rco2	**15**	0,00	85,89				**15**	0,00	1,07
rco3	**22**	0,00	404,13				**22**	0,00	141,86
rco4	30	13,33	3600,00				**29**	3,45	3600,00
rco5	37	13,51	3600,00				**36**	2,78	3600,00
shapes2	**14**	0,00	2,10				**14**	0,00	1,09
shapes4	**25**	0,00	2431,50				**25**	4,00	3600,00
shapes5	**31**	9,68	3600,00				**31**	12,86	3600,00
shapes7	**42**	28,57	3600,00				45	24,44	3600,00
shapes9	**49**	32,65	3600,00				54	29,63	3600,00
shapes15	**62**	35,48	3600,00				-	100,00	3600,00
shirts1-2	**13**	0,00	0,03				**13**	0,00	0,02
shirts2-4	**17**	0,00	177,29				**17**	0,00	47,77
shirts3-6	**24**	0,00	3558,46				**24**	0,00	497,68
shirts4-8	**33**	15,15	3600,00				**33**	6,06	3600,00
shirts5-10	**41**	14,63	3600,00				-	100,00	3600,00
three	**6**	0,00	0,11	**6**	0,00	0,01	**6**	0,00	0,01
threep2	10	0,00	0,32	**9,33**	0,00	0,80	10	0,00	0,01
threep2w9	**8**	0,00	0,86	**8**	0,00	4,80	**8**	0,00	0,01
threep3	14	0,00	0,58	**13,53**	16,26	3600,00	14	0,00	0,01
threep3w9	12	0,00	5,57	**11**	0,00	2144,50	12	0,00	0,06
Média	-	3,88	1088,70	-	4,62	1124,84	-	-	-
Des. Padrão	-	8,21	1590,56	-	8,04	1637,49	-	-	-
Média	-	7,69	1804,32	-	-	-	-	9,03	1489,29
Des. Padrão	-	10,53	1733,23	-	-	-	-	22,19	1725,86

Tabela 3.4 – Resultados para o caso moderado.

					Largura por Cenário									
Instância	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)	L_0	L^1_1	L^2_1	L^3_1	L^4_1	L^5_1	L^6_1	L^7_1	L^8_1	L^9_1
blazewicz1	9	192,19	15,85	21600,00	7	0	0	0	0	5	5	6	8	10
fu5	6	902,50	22,24	21600,00	20	0	0	0	0	3	4	4	4	8
fu6	7	1142,37	16,32	21600,00	25	0	3	0	0	0	6	5	9	13
fu7	8	1305,06	15,65	21600,00	29	0	0	0	0	3	9	4	5	13
fu8	10	1441,62	11,53	21600,00	25	0	0	0	0	12	15	3	19	19
poly1c	20	810,00	33,49	21600,00	18	0	0	1	0	0	2	3	5	6
rco1	9	196,88	0,00	14990,29	12	0	0	0	1	0	1	0	2	4
shapes2	9	840,00	31,55	21600,00	15	0	0	0	0	3	7	7	10	11
shirts1-2	16	677,50	22,69	21600,00	13	0	0	0	0	3	4	5	5	7
three	4	57,53	0,00	557,16	7	0	0	0	0	0	3	0	2	3
threep2	6	94,28	0,00	291,66	10	0	0	0	0	1	4	5	6	8
threep2w9	6	106,59	0,00	4572,56	8	0	0	0	0	4	4	3	4	6
threep3	9	133,00	0,00	964,29	13	0	0	0	0	4	6	7	9	11
threep3w9	10	135,28	0,00	2048,33	11	0	0	0	0	2	6	5	5	8
Média	-	-	12,09	14016,02	-	-	-	-	-	-	-	-	-	-
Des. Padrão	-	-	12,32	9740,22	-	-	-	-	-	-	-	-	-	-

10,53% e 1733,23 segundos, respectivamente.

3.4.3 Problema com incertezas

A análise do algoritmo *branch-and-cut* proposto e das incertezas no 2ISP-DU consideram a resolução de instâncias da Tabela 3.1, com os 9 cenários definidos na Tabela 3.2, incluindo o cálculo do EVPI e VSS para um estudo da importância em se considerar incertezas no problema. Realizam-se também testes sobre instâncias com um número maior de cenários para avaliar a performance do algoritmo exato e o impacto nas soluções do problema. As Tabelas 3.4 a 3.7 contêm os resultados considerando as instâncias com 9 cenários para os quatro casos (moderado, equiprovável, otimista e pessimista). Em todos os experimentos do problema com incertezas, limitou-se o tempo do *solver* em 21600 segundos (6 horas) para a resolução de cada instância.

Observando resultados individuais nas Tabelas 3.4 a 3.7, o número de instâncias com soluções ótimas, o *gap* médio e o tempo médio do algoritmo exato são, respectivamente: 6, 12,09% e 14016,02 segundos, com desvio padrão de 12,32% e 9740,22 segundos (no caso moderado); 5, 11,94% e 14618,01 segundos, com desvio padrão de 10,99% e 9797,73 segundos (no caso equiprovável); 5, 7,66% e 14303,78 segundos, com desvio padrão de 7,08% e 10192,64 segundos (no caso otimista); e 5, 13,57% e 14159,87 segundos, com desvio padrão de 13,56% e 10376,57 segundos (no caso pessimista). Nota-se que uma instância a mais foi resolvida na otimalidade no caso moderado (instância rco1). Esse resultado foi alcançado devido às soluções geradas pela heurística e inseridas durante a otimização. Em relação ao *gap* médio, o caso otimista apresenta melhor valor, seguido pelo caso equiprovável, enquanto os piores *gaps* surgem no caso pessimista, indicando que o problema torna-se mais difícil quando há probabilidade de ocorrência de cenários com maior demanda para os itens. Em termos de tempo computacional, a

Tabela 3.5 – Resultados para o caso equiprovável.

					Largura por Cenário									
Instância	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)	L_0	L^1_1	L^2_1	L^3_1	L^4_1	L^5_1	L^6_1	L^7_1	L^8_1	L^9_1
blazewicz1	9	197,50	21,07	21600,00	11	0	0	0	0	1	3	1	4	4
fu5	6	918,35	16,90	21600,00	20	0	0	0	0	4	4	4	5	8
fu6	8	1266,75	14,00	21600,00	27	0	0	0	0	7	4	5	11	11
fu7	9	1355,42	14,95	21600,00	28	0	0	0	0	10	6	6	8	16
fu8	10	1361,57	10,23	21600,00	29	0	0	0	0	4	5	8	9	15
poly1c	19	840,00	33,34	21600,00	19	0	0	0	0	1	3	0	3	5
rco1	10	214,99	13,93	21600,00	12	0	0	0	0	2	0	2	4	6
shapes2	8	826,53	28,22	21600,00	15	0	0	0	0	1	4	7	10	12
shirts1-2	15	639,93	14,58	21600,00	13	0	0	0	0	0	4	3	5	6
three	4	64,18	0,00	137,20	7	0	0	0	0	3	3	2	2	3
threep2	6	101,49	0,00	4732,56	11	0	0	0	0	2	3	4	5	7
threep2w9	7	104,99	0,00	159,66	8	0	0	0	0	2	4	3	5	8
threep3	10	142,32	0,00	1123,28	14	0	0	0	0	3	8	8	8	11
threep3w9	9	130,49	0,00	4099,41	10	0	0	0	0	2	7	4	6	8
Média	-	-	11,94	14618,01	-	-	-	-	-	-	-	-	-	-
Des. Padrão	-	-	10,99	9797,73	-	-	-	-	-	-	-	-	-	-

Tabela 3.6 – Resultados para o caso otimista.

					Largura por Cenário									
Instância	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)	L_0	L^1_1	L^2_1	L^3_1	L^4_1	L^5_1	L^6_1	L^7_1	L^8_1	L^9_1
blazewicz1	9	133,80	6,73	21600,00	4	0	4	3	4	7	5	8	10	10
fu5	6	717,82	8,81	21600,00	14	0	0	0	6	8	10	10	14	14
fu6	6	767,98	14,99	21600,00	14	0	0	8	4	10	14	11	24	24
fu7	9	1016,50	15,03	21600,00	14	0	10	14	10	14	24	16	28	28
fu8	10	978,31	14,04	21600,00	14	0	1	14	10	24	23	16	30	30
poly1c	20	674,80	21,34	21600,00	13	0	3	2	1	8	10	5	10	12
rco1	9	149,02	6,34	21600,00	2	0	7	6	7	10	12	11	12	13
shapes2	9	651,60	11,85	21600,00	12	0	2	0	2	7	11	11	13	13
shirts1-2	16	564,60	8,08	21600,00	12	0	1	1	1	4	1	5	7	9
three	4	53,13	0,00	25,89	6	0	1	1	1	4	2	1	4	4
threep2	6	65,38	0,00	178,57	4	0	3	6	3	7	11	11	12	14
threep2w9	7	72,58	0,00	678,45	4	0	1	4	4	6	8	7	9	10
threep3	10	102,27	0,00	1248,22	6	0	5	9	8	11	14	13	16	19
threep3w9	9	96,62	0,00	3721,75	4	0	6	4	5	9	11	10	13	14
Média	-	-	7,66	14303,78	-	-	-	-	-	-	-	-	-	-
Des. Padrão	-	-	7,08	10192,64	-	-	-	-	-	-	-	-	-	-

Tabela 3.7 – Resultados para o caso pessimista.

					Largura por Cenário									
Instância	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)	L_0	L^1_1	L^2_1	L^3_1	L^4_1	L^5_1	L^6_1	L^7_1	L^8_1	L^9_1
blazewicz1	9	200,25	20,86	21600,00	12	0	0	0	0	0	0	0	1	2
fu5	6	1035,12	13,99	21600,00	24	0	0	0	0	0	0	0	4	4
fu6	7	1378,64	11,27	21600,00	28	0	0	0	0	0	0	2	10	10
fu7	9	1608,16	13,35	21600,00	38	0	0	0	0	0	0	0	4	6
fu8	9	1520,76	11,14	21600,00	33	0	0	0	0	0	0	0	4	11
poly1c	20	981,60	37,45	21600,00	24	0	0	0	0	0	0	0	0	1
rco1	9	223,35	14,34	21600,00	13	0	0	0	0	2	0	0	2	2
shapes2	9	974,00	36,59	21600,00	23	0	0	0	0	0	0	0	1	2
shirts1-2	15	784,80	30,99	21600,00	18	0	0	0	0	0	0	0	2	2
three	3	66,08	0,00	60,83	7	0	0	0	0	1	0	2	2	3
threep2	7	116,90	0,00	180,85	14	0	0	0	0	0	0	0	2	4
threep2w9	6	116,01	0,00	181,16	11	0	0	0	0	0	0	0	1	3
threep3	9	169,61	0,00	749,58	20	0	0	0	0	0	0	2	3	6
threep3w9	9	159,30	0,00	2665,69	15	0	0	0	0	0	0	0	2	4
Média	-	-	13,57	14159,87	-	-	-	-	-	-	-	-	-	-
Des. Padrão	-	-	13,56	10376,57	-	-	-	-	-	-	-	-	-	-

média, em cada caso, ficou em torno dos 14000 segundos, indicando que mesmo instâncias com poucos cenários e itens por cenário podem ser muito difíceis de serem resolvidas otimamente (por exemplo, a instância blazewicz1).

Embora os cenários tenham sido gerados de forma independente, analisa-se o que acontece entre os casos nas Tabelas 3.4 a 3.7. Ao comparar o caso moderado com o equiprovável, houve redução no *gap* médio de 0,15 pontos percentuais, ocorrendo a redução do *gap* em todas as instâncias (exceto na blazewicz1 e rco1). O tempo computacional, por outro lado, apresentou um aumento percentual de 4,30%. Do caso moderado para o otimista, o *gap* médio reduziu 4,43 pontos percentuais, com redução do *gap* exceto para as instâncias fu8 e rco1. O tempo computacional médio teve um aumento de 2,05%. Por outro lado, na comparação entre o caso moderado com o pessimista, houve um aumento no *gap* médio de 1,48 pontos percentuais, ocorrendo a redução do *gap* somente nas instâncias fu5, fu6, fu7 e fu8. O tempo computacional médio também teve um aumento de 1,03%. Nota-se que apenas no caso moderado foi possível resolver 6 instâncias na otimalidade, enquanto nos demais, esse número reduz-se para 5 de 14 instâncias. Observa-se ainda que, em todos os casos, as instâncias com maior *gap* são as poly1c e shapes2, justificando-se pela quantidade de itens e/ou geometria dos itens.

Ao analisar a largura da faixa nos diferentes cenários das instâncias nas Tabelas 3.4 a 3.7, para o caso moderado, nota-se que a largura inicial de faixa é capaz de atender do cenário 1 (Baixa-Baixa) ao 4 (Média-Baixa) sem qualquer necessidade de largura de faixa adicional (exceto para as instâncias fu6, poly1c e rco1). Esse resultado também é observado no caso equiprovável, em que uma largura de faixa adicional passa a ser requisitada do cenário 5 em diante. No caso otimista, apenas no cenário 1 é que não se precisa de uma largura de faixa adicional em qualquer uma das instâncias. Isso se justifica pela alta probabilidade associada aos

cenários com baixa demanda para os itens. Por outro lado, no caso pessimista, como os cenários com alta demanda tem probabilidade maior de ocorrerem, a largura inicial da faixa é capaz de atender totalmente os cenários de 1 a 7 (exceto para algumas poucas instâncias). Além disso, nos demais cenários (8 e 9 do caso pessimista), a largura de faixa adicional é menor se comparada aos mesmos cenários dos demais casos (moderado, equiprovável e otimista). Em geral, observa-se que as maiores larguras de faixa adicionais ocorrem para os cenários cuja demanda dos itens originam dos grupos Médio e Alto.

Uma outra análise do problema estocástico diz respeito ao valor esperado da informação perfeita (isto é, o EVPI). A Tabela 3.8 contém os resultados para 6 instâncias dos casos moderado, equiprovável, otimista e pessimista, em que foi possível obter a solução ótima RP (exceto para a instância shirts1-2) nas Tabelas 3.4 a 3.7. O algoritmo exato proposto conseguiu obter a solução ótima dos problemas *wait-and-see*, para cada cenário s (isto é, WS_s^*), de todas as instâncias em análise. Para cada instância e caso, as três últimas linhas contêm o valor WS, o EVPI e a solução RP (isto é, do modelo equivalente determinístico). Apresenta-se também o EVPI relativo (em porcentagem), que é calculado como $100 \times (\text{EVPI}/\text{RP})$.

Na Tabela 3.8, as soluções dos problemas *wait-and-see* acompanham a probabilidade de ocorrência de cada cenário para cada caso na Tabela 3.2. Em geral, para cada caso (moderado, equiprovável, otimista e pessimista), quanto maior a probabilidade de ocorrência do cenário, maior tem sido o valor da solução do problema *wait-and-see*. Na instância shirts1-2, embora a solução RP não seja ótima, o EVPI relativo ainda é relativamente alto, isto é, de 82,18% para o caso moderado, 85,53% para o caso equiprovável, 71,73% para o caso otimista e 74,04% para o caso pessimista. Nas demais instâncias, cuja solução RP é ótima em todos os casos, o EVPI relativo também é alto. Ou seja, na instância three, o EVPI relativo é de 80,89%, 85,86%, 72,73% e 69,28% para os casos moderado, equiprovável, otimista e pessimista, respectivamente. Na instância threep2, o EVPI relativo é de 81,95%, 86,84%, 81,04% e 68,45%, respectivamente, enquanto que para a instância threep2w9, esses valores são 81,60%, 86,03%, 79,30% e 68,40%. Para a instância threep3, o EVPI relativo é de 81,99% para o caso moderado, 86,24% para o caso equiprovável, 80,62% para o caso otimista e 69,76% para o caso pessimista, enquanto para a instância threep3w9 esses valores são, respectivamente, 81,95%, 86,08%, 81,15% e 68,57%.

Pela análise do EVPI na Tabela 3.8, nota-se que aleatoriedade é importante dentro do problema em estudo, principalmente pelos valores elevados do EVPI em cada caso. Na média geral, o EVPI relativo é de 81,76% (com desvio padrão de 0,47%) para o caso moderado, 86,10% (com desvio padrão de 0,44%) para o caso equiprovável, 77,76% (com desvio padrão de 4,35%) para o caso otimista e 69,75% (com desvio padrão de 2,17%) para o caso pessimista. Embora o caso pessimista tenha apresentado o menor valor em comparação aos demais, esse valor ainda é alto (acima dos 60%). Com isso, caso o tomador de decisões não tenha conhecimento perfeito e preciso das demandas dos itens, é altamente vantajoso resolver um modelo de programação estocástica para decidir sobre qual área de faixa comprar inicialmente.

Tabela 3.8 – Análise EVPI para instâncias com 9 cenários.

	Moderado	Equiprovável	Otimista	Pessimista	Moderado	Equiprovável	Otimista	Pessimista
Cenário	WS_s^* para shirts1-2*				WS_s^* para three			
1	33,75	79,85	259,20	7,20	2,62	4,66	22,68	0,32
2	97,50	80,06	140,40	23,40	7,88	8,17	13,23	1,26
3	48,75	59,89	46,80	32,40	4,59	8,15	4,41	2,52
4	97,50	86,74	140,40	23,40	9,19	4,67	13,23	1,26
5	210,00	87,05	86,40	70,20	18,38	11,72	9,45	7,56
6	120,00	106,75	23,40	140,40	13,12	11,68	2,52	13,23
7	60,00	106,46	57,60	57,60	4,59	10,48	4,41	5,67
8	127,50	113,42	30,60	183,60	11,81	10,51	3,15	17,01
9	63,75	113,12	10,20	367,20	6,56	11,64	1,05	37,80
WS	120,70	92,59	159,62	203,71	10,99	9,08	14,49	20,30
RP	677,50	639,93	564,60	784,80	57,53	64,18	53,13	66,08
EVPI	556,80	547,34	404,98	581,09	46,54	55,10	38,64	45,78
EVPI (%)	82,18	85,53	71,73	74,04	80,89	85,86	72,73	69,40
	WS_s^* para threep2				WS_s^* para threep2w9			
1	2,62	3,49	15,12	0,42	2,53	7,49	19,44	0,40
2	10,50	4,67	13,23	2,21	6,75	10,51	12,15	2,83
3	6,56	12,81	6,30	6,30	6,75	11,98	6,48	6,48
4	13,12	10,51	13,23	3,15	13,50	12,01	19,44	3,64
5	28,88	15,23	10,40	12,29	40,50	15,07	12,15	10,94
6	18,38	16,35	4,72	26,46	20,25	18,01	4,86	26,73
7	9,84	17,47	9,45	8,82	9,28	16,47	8,91	8,91
8	21,00	18,68	5,04	30,24	20,25	19,52	5,26	29,16
9	11,81	20,96	1,89	68,04	11,81	20,96	1,89	68,04
WS	17,02	13,35	12,40	36,88	19,62	14,67	15,02	36,66
RP	94,28	101,49	65,38	116,9	106,59	104,99	72,58	116,01
EVPI	77,26	88,14	52,98	80,02	86,97	90,32	57,56	79,35
EVPI (%)	81,95	86,84	81,04	68,45	81,60	86,03	79,30	68,40
	WS_s^* para threep3				WS_s^* para threep3w9			
1	4,59	4,66	22,68	0,42	4,22	5,99	19,44	0,81
2	10,50	12,84	20,79	3,46	11,81	12,01	24,30	1,62
3	8,53	16,30	9,45	7,56	9,28	14,97	6,48	8,10
4	17,06	16,35	26,46	3,46	18,56	15,01	21,87	4,46
5	44,62	19,92	16,06	13,23	43,88	18,08	15,80	15,80
6	24,94	25,69	6,30	37,80	28,69	25,52	6,07	36,45
7	13,12	25,62	11,97	13,23	13,50	20,96	11,34	12,15
8	28,88	25,69	6,93	43,47	27,00	24,02	6,88	41,31
9	15,75	29,11	2,62	94,50	16,03	26,95	2,43	92,34
WS	23,95	19,58	19,82	51,30	24,42	18,17	18,21	50,07
RP	133,00	142,32	102,27	169,61	135,28	130,49	96,62	159,30
EVPI	109,05	122,74	82,45	118,31	110,86	112,32	78,41	109,23
EVPI (%)	81,99	86,24	80,62	69,76	81,95	86,08	81,15	68,57

Tabela 3.9 – Análise VSS para instâncias com 9 cenários.

Instância	Obj. EVV	VSS	Dev. (%)	Obj. EVV	VSS	Dev. (%)
	Moderado			Equiprovável		
fu5*	1064,00	161,50	15,18	1064	145,65	13,69
rco1*	225,00	28,12	12,50	225	10,01	4,45
shirts1-2*	680,00	2,50	0,37	680	40,07	5,89
three	70,00	12,47	17,81	70	5,82	8,31
threep2	126,00	31,72	25,17	126	24,51	19,45
threep2w9	126,00	19,41	15,40	126	21,01	16,67
threep3	168,00	35,00	20,83	175	32,68	18,67
threep3w9	171,00	35,72	20,89	162	31,51	19,45
Média	-	-	16,02	-	-	13,32
Des. Padrão	-	-	7,49	-	-	6,26
	Otimista			Pessimista		
fu5*	1064	346,18	32,54	1064	28,88	2,71
rco1*	225	75,98	33,77	225	1,65	0,73
shirts1-2*	680	115,4	16,97	680	-104,8	-15,41
three	70	16,87	24,10	70	3,67	5,24
threep2	126	60,62	48,11	126	9,1	7,22
threep2w9	126	53,42	42,40	126	9,99	7,93
threep3	175	72,73	41,56	175	5,39	3,08
threep3w9	162	65,38	40,36	171	11,7	6,84
Média	-	-	34,98	-	-	4,82*
Des. Padrão	-	-	10,35	-	-	2,70*

A Tabela 3.9 contém os resultados relacionados ao ganho em se resolver o problema estocástico em comparação a um problema de valor esperado. Ou seja, analisa-se o VSS para 8 instâncias, embora para três delas (fu5, rco1 e shirts1-2) a solução RP não é ótima dentro do tempo limite de 21600 segundos, conforme apresentado nas Tabelas 3.4 a 3.7. Para o problema de valor esperado (isto é, EV), assume-se o cenário de referência como sendo o pior cenário em termos de demanda dos itens na Tabela 3.2. A solução do EV é, então, utilizada para obter o EVV, a respeito do custo de se ignorar a aleatoriedade do problema. Todas as soluções EV e EVV são ótimas, ou seja, as instâncias foram resolvidas na otimalidade pelo algoritmo *branch-and-cut* proposto considerando o tempo limite definido. Além disso, apresenta-se o desvio relativo (isto é, a porcentagem de piora) da solução do EVV em comparação com a solução RP, dado por $100 \times (\text{EVV} - \text{RP})/\text{EVV}$.

A Tabela 3.9 traz os valores do EVV, VSS e o desvio relativo (isto é, a redução percentual) da solução EVV sobre a RP. Nota-se que em todos os casos, o desvio relativo é positivo, exceto para a instância shirts1-2, no caso pessimista, pois neste caso a solução RP não é ótima e, assim, no pior caso, a solução EVV deveria ser igual a RP, com desvio relativo nulo. No caso moderado, exceto para a instância shirts1-2 cujo desvio relativo foi inferior a 0,5%, o desvio relativo nas demais instâncias foi superior a 12%, chegando a 25,17% na instância threep2. Assim, ocorre uma redução satisfatória no custo quando se decide pela resolução do problema estocástico ao

invés de se considerar um problema de valor esperado. No caso equiprovável, o desvio relativo varia entre 4,45% e 8,31% para as instâncias rco1, shirts1-2 e three, enquanto permanece superior aos 13% para as demais instâncias, chegando a 19,45% para as instâncias threep2 e threep3w9.

Observando ainda a Tabela 3.9, o caso otimista é o que apresenta os maiores desvios relativos, sempre superiores aos 15% para qualquer instância, chegando a 48,11% na instância threep2. Por outro lado, o caso pessimista é o que possui os menores desvios percentuais, todos inferiores a 9%. Ainda assim, para metade das instâncias no caso pessimista, o desvio foi superior a 5%, o que certamente impacta no custo final do problema. A média e o desvio padrão do desvio relativo são 16,02% e 7,49%, 13,32% e 6,26%, 34,98% e 10,35%, e 4,82% e 2,70% para os casos moderado, equiprovável, otimista e pessimista (excluindo nesse último a instância shirts1-2), respectivamente. Esses resultados apontam que a resolução do problema estocástico deveria ser considerada quando se busca por decisões de custo mínimo, mesmo que a solução ótima do modelo de programação estocástica possa exigir bastante tempo computacional para ser obtida.

A Tabela 3.10 apresenta os resultados para instâncias com 20, 40, 60 e 80 cenários. O objetivo desses testes é mensurar a capacidade (escalabilidade) do algoritmo proposto para resolver instâncias do problema com um maior número de cenários. Na tabela são apresentados o número médio de itens por cenário, a solução final obtida (isto é, o valor da função objetivo), o *gap* percentual retornado pelo *solver* e o tempo computacional em segundos. Os cenários foram gerados seguindo a Tabela 3.2, porém com probabilidade igual de ocorrência (isto é, são cenários equiprováveis).

Nos resultados da Tabela 3.10, o número de instâncias com solução ótima, o *gap* médio e o tempo computacional médio do algoritmo proposto são, respectivamente: para 20 cenários, 5, 12,77% e 15930,43 segundos (com desvio padrão de 11,99% e 9038,81 segundos); para 40 cenários, 3, 16,75% e 19372,58 segundos (com desvio padrão de 12,14% e 5572,99 segundos); para 60 cenários, 2, 18,85% e 18989,36 segundos (com desvio padrão de 10,94% e 6682,96 segundos); e, para 80 cenários, 1, 19,30% e 20171,55 segundos (com desvio padrão de 10,78% e 5344,79 segundos). Observa-se que o número de instâncias resolvidas a otimalidade reduz a medida que o número de cenários cresce. Por outro lado, o *gap* médio aumenta a medida que o número de cenários cresce, com aumento de 3,97% pontos percentuais de 20 para 40 cenários, de 2,11% pontos percentuais de 40 para 60 cenários, e de 0,45% pontos percentuais de 60 para 80 cenários.

O tempo computacional médio requerido pelo algoritmo proposto, como apresentado na Tabela 3.10, teve um aumento de 21,61% ao passar de 20 para 40 cenários, uma leve redução de 1,98% ao passar de 40 para 60 cenários, e um aumento de 6,23% ao passar de 60 para 80 cenários (enquanto que a tendência do desvio padrão foi diminuir). Observa-se ainda que as instâncias com maior *gap* são as blazewicz1, poly1c e shapes2, com valores próximo de 30%. Além disso, para algumas instâncias há a tendência do *gap* crescer a medida que o número de cenários cresce, como é caso da blazewicz1, threep2 e threep2w9. Esse comportamento pode não

Tabela 3.10 – Resultados para instâncias com até 80 cenários.

	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)	I^s_{avg}	Obj.	*gap* (%)	Tempo (s)
Instância	20 cenários				40 cenários			
blazewicz1	8	189,75	29,05	21600,00	9	197,81	29,62	21600,00
fu5	6	854,05	16,91	21600,00	6	889,20	20,01	21600,00
fu6	7	1211,25	16,55	21600,00	7	1201,27	17,64	21600,00
fu7	8	1285,35	15,08	21600,00	8	1285,35	15,67	21600,00
fu8	10	1365,15	11,00	21600,00	10	1319,55	12,31	21600,00
poly1c	19	843,00	34,40	21600,00	19	847,50	34,93	21600,00
rco1	8	184,13	13,44	21600,00	9	207,19	23,17	21600,00
shapes2	8	827,00	29,50	21600,00	9	855,50	31,56	21600,00
shirts1-2	14	628,00	12,90	21600,00	15	689,50	22,29	21600,00
three	3	57,23	0,00	170,87	3	57,14	2,76	21600,00
threep2	6	96,42	0,00	441,17	6	99,75	0,00	1530,63
threep2w9	6	95,62	0,00	1189,18	6	96,41	0,00	13960,60
threep3	9	139,13	0,00	8190,37	9	136,85	0,00	18124,94
threep3w9	9	126,23	0,00	18634,46	9	138,60	24,51	21600,00
Média	-	-	12,77	15930,43	-	-	16,75	19372,58
Des. Padrão	-	-	11,99	9038,81	-	-	12,14	5572,99
	60 cenários				80 cenários			
blazewicz1	9	196,14	30,02	21600,00	9	201,38	30,21	21600,00
fu5	6	874,07	18,25	21600,00	6	908,20	19,87	21600,00
fu6	7	1170,45	17,04	21600,00	7	1202,70	16,59	21600,00
fu7	8	1299,68	15,50	21600,00	8	1291,05	15,14	21600,00
fu8	10	1341,49	11,83	21600,00	10	1388,19	13,12	21600,00
poly1c	19	884,03	37,11	21600,00	20	880,00	36,65	21600,00
rco1	9	201,01	20,87	21600,00	9	206,25	20,34	21600,00
shapes2	9	843,06	29,66	21600,00	9	857,75	32,71	21600,00
shirts1-2	15	656,03	19,82	21600,00	15	664,00	20,44	21600,00
three	4	59,85	0,00	1311,91	3	59,98	0,00	1601,64
threep2	6	99,76	0,00	5339,16	7	102,02	2,32	21600,00
threep2w9	6	107,11	13,76	21600,00	6	104,68	22,88	21600,00
threep3	9	146,31	32,05	21600,00	9	147,79	10,75	21600,00
threep3w9	9	135,91	18,04	21600,00	10	149,29	29,23	21600,00
Média	-	-	18,85	18989,36	-	-	19,30	20171,55
Des. Padrão	-	-	10,94	6682,96	-	-	10,78	5344,79

ter ocorrido nas demais instâncias devido a qualidade das soluções geradas pela heurística. De forma geral, o algoritmo proposto tem conseguido resolver instâncias com um número razoável de cenários, principalmente pela boa capacidade da heurística em gerar soluções de qualidade.

3.5 Considerações finais

O problema de corte de itens irregulares em faixa bidimensional com demanda incerta foi resolvido através de um algoritmo *branch-and-cut* que considera um modelo de programação estocástica de dois estágios e uma heurística baseada em busca em vizinhança variável. O objetivo do modelo é determinar o tamanho da faixa a comprar, que seja de menor custo, para atender a demanda de itens que será conhecida no futuro. No primeiro estágio, o modelo obtém o custo total da faixa, que depende do custo esperado do modelo de segundo estágio, em conformidade com os cenários utilizados. O problema traz uma dificuldade intrínseca relacionada a não sobreposição de itens, sendo utilizado os métodos *no-fit raster* e *inner-fit raster* para tal fim.

O algoritmo proposto resolve 25 das 44 instâncias do problema sem incertezas na demanda na otimalidade, com um *gap* médio de 8,14% e tempo médio de 1847,99 segundos. Ao mesmo tempo, na presença de incertezas, o algoritmo foi testado em instâncias com uma árvore de cenários representando as realizações das demandas. A primeira análise do algoritmo exato sobre quatro casos de cenários (isto é, moderado, equiprovável, otimista e pessimista) mostra que a solução varia conforme as probabilidades dos casos e cenários. Por outro lado, para o problema com incertezas, para as instâncias com 9 cenários, aproximadamente 35% delas foram resolvidas na otimalidade, com os piores *gaps* para o caso pessimista e o tempo médio de execução próximo dos 14000 segundos em cada caso. Os resultados para as instâncias com maior quantidade de cenários mostram conclusões similares. As análises de valor esperado da informação perfeita mostram que o valor relativo entre as soluções *wait-and-see* e do problema estocástico é alto, com o EVPI relativo médio variando de 69,75% (para o caso pessimista) a 86,10% (para o caso equiprovável). De forma similar, nas análises sobre o problema de valor esperado, o desvio relativo médio varia entre 4,82% (para o caso pessimista) a 34,98% (para o caso otimista), indicando que ignorar a aleatoriedade dos parâmetros na escolha de uma decisão pode resultar em custos elevados para o problema.

CAPÍTULO

4

MODELO DE PROGRAMAÇÃO ESTOCÁSTICA PARA O PROBLEMA DA MOCHILA COM INCERTEZAS NOS DEFEITOS DA PLACA

O presente capítulo lida com o problema da mochila bidimensional considerando o corte de itens irregulares em uma placa com defeitos. Enquanto os defeitos só são conhecidos no momento do corte, deve-se selecionar inicialmente quais itens produzir a partir do corte da placa. Os itens finais não podem ter qualquer parte com defeito e o objetivo é maximizar o lucro com o corte da placa e produção dos itens. Propõe-se um modelo de programação estocástica de dois estágios com recurso, que faz uso de um conjunto discreto de cenários com a realização dos defeitos na placa. As decisões de primeiro estágio envolvem selecionar os itens para o corte e possível produção, enquanto as decisões de segundo estágio consideram o posicionamento dos itens na placa com a realização dos defeitos, podendo, então, haver o cancelamento e a não produção de itens selecionados. Considera-se ainda a extensão desse modelo para incluir uma medida de aversão ao risco, com o objetivo de obter soluções robustas. Realizamos testes computacionais sobre instâncias adaptadas da literatura que consideram três tipos de defeitos, oito cenários e quatro casos de probabilidades de ocorrência dos cenários. Os testes ainda avaliam o impacto das incertezas no problema a partir da análise do valor esperado da informação perfeita e do valor da solução estocástica. Nos resultados foi possível obter soluções totalmente robustas com o modelo averso ao risco, porém pagando o preço de reduções percentuais no lucro de até 27,7% na média.

Alguns resultados iniciais desse capítulo foram publicados como artigo completo e apresentados no *LIII Simpósio Brasileiro de Pesquisa Operacional* (SOUZA QUEIROZ; ANDRETTA, 2021). Todo o capítulo está sendo submetido para um periódico internacional.

4.1 Introdução

Nas empresas de corte de couro, de tecido e metal-mecânica, uma placa (isto é, uma peça grande de couro, de tecido ou de metal) está disponível para ser cortada e obter itens, que podem ter formato irregular. Esses itens são usados para a produção de roupas, bolsas, calçados, artefatos de esporte, entre outros (BENNELL; OLIVEIRA, 2009). A placa pode ainda apresentar defeitos em algumas partes, seja oriundos da extração, manuseio ou transporte, inviabilizando a obtenção de itens sobre essas partes (BALDACCI *et al.*, 2014).

Nesse trabalho, considera-se o corte de itens irregulares a partir de uma placa retangular que pode apresentar defeitos. Assume-se que a placa é comprada com certa antecendência, devido a questões de preparo e transporte por parte do fornecedor. A placa possui dimensões e características de material conhecidas. Todavia, os defeitos na placa são identificados somente após a entrega e quando ela adentra o estágio de corte, momento em que itens previamente selecionados podem ter a sua produção cancelada. Deseja-se um plano de corte (para a placa) que dê o máximo lucro líquido, que depende do lucro obtido com a produção e entrega de itens menos o custo de cancelamento de itens, que foram selecionados mas não produzidos. O plano de corte considera os itens selecionados para serem obtidos da placa e a realização dos possíveis defeitos através de cenários, podendo ser necessário o cancelamento de itens quando do efetivo corte da placa. Dessa forma, os defeitos são tratados como dados incertos. Esse é um problema do tipo mochila bidimensional com itens irregulares e incertezas sobre os defeitos da placa (*Two-dimensional Irregular Knapsack Problem with Defects Uncertainty - 2IKP-DU*) .

Problemas de corte de itens irregulares vêm sendo estudados na literatura de problemas de corte e empacotamento, com um número de contribuições relativamente menor comparado aos problemas com itens regulares, especialmente aqueles com formato retangular (WÄSCHER; HAUSSNER; SCHUMANN, 2007). Restrições comuns em problemas dessa natureza dizem respeito a garantir a não sobreposição entre itens e que o posicionamento dos itens não venha a extrapolar as dimensões da placa. Devido à geometria irregular dos itens, torna-se necessário o uso de ferramentas adicionais para que sejam garantidas essas restrições. Algumas ferramentas incluem a discretização dos itens em matrizes de pontos (método *raster*), o cálculo da posição relativa entre os itens, com o uso de funções de distância (*phi-functions*), a construção de polígonos de obstrução entre pares de itens (*no-fit polygon*) e a verificação da interseção entre segmentos e de inclusão de pontos entre itens (trigonometria direta) (BENNELL; OLIVEIRA, 2008).

Com relação ao problema da mochila com itens irregulares, Scheithauer e Terno (1993) desenvolveram modelos de programação linear inteira mista para lidar com polígonos convexos e polígonos quaisquer. Martins e Tsuzuki (2010) desenvolveram uma heurística de recozimento simulado para o problema contendo placas retangulares e irregulares. As regiões para posicionar itens são obtidas a partir do *no-fit polygon*. Valle *et al.* (2012) desenvolveram heurísticas baseadas em GRASP, com a resolução das variantes em que existe apenas uma cópia de cada item e

quando há ilimitadas cópias de cada item. O posicionamento dos itens é feito usando o *no-fit polygon*. Mundim e Queiroz (2012) desenvolveram uma busca local baseada no recozimento simulado e integraram no GRASP de Valle *et al.* (2012). Silveira (2013) utilizou um algoritmo genético cujos cromossomos representam a sequência em que os itens devem ser posicionados na placa. Os itens são posicionados por uma heurística de agrupamento que faz uso do *no-fit polygon*.

Em Dalalah, Khrais e Bataineh (2014), o problema da mochila considera placas retangulares e irregulares. Esses autores desenvolveram uma heurística construtiva que posiciona os itens conforme a ocupação da envoltória convexa. Baldacci *et al.* (2014) resolveram o problema com placa irregular que surge em uma indústria de couro. A placa é discretizada em uma matriz de pontos e os autores apresentaram um modelo de programação linear inteira, uma relaxação lagrangeana e heurísticas obtidas a partir da relaxação que propuseram. Mundim *et al.* (2018) desenvolveram uma heurística geral para problemas de corte de itens irregulares, como o da mochila, empacotamento em recipientes e de corte de estoque. Os autores propuseram diferentes regras de posicionamento, algumas delas baseadas na *bottom-left*, e conseguiram melhorar grande parte das soluções apresentadas por outros trabalhos da literatura. Souza Queiroz e Andretta (2020b) propuseram duas heurísticas, a primeira baseada em um algoritmo genético de chaves aleatórias viciadas e a outra é uma busca em vizinhança variável. As heurísticas consideram a solução codificada através de um vetor de números inteiros e o posicionamento dos itens observa o *no-fit polygon* discretizado e três regras inspiradas na *bottom-left*. Román (2020) considerou o posicionamento de um subconjunto de itens de máximo valor e que respeite o peso máximo que pode ser cortado da placa. O autor assumiu que os itens podem ser rotacionados e a placa pode ser irregular e apresentar defeitos. O autor desenvolveu várias heurísticas para o problema, incluindo um algoritmo evolutivo, enquanto a não sobreposição de itens é verificada por uma biblioteca da linguagem de programação Python.

Alguns estudos têm considerado defeitos e zonas de qualidade nas placas de problemas de corte de itens irregulares. Em Chung, Scott III e Hillman (1990) há uma heurística híbrida que aloca os itens em regiões da placa definidas de acordo com a posição dos defeitos. Heistermann e Lengauer (1995) lidaram com um caso real encontrado na indústria de couro, em que a placa é irregular e contém zonas de qualidade e defeitos. A heurística dos autores considera um conjunto de itens candidatos e tenta posicioná-los sem gerar sobreposição caso seja possível, senão posiciona apenas o item que resulta na melhor ocupação da placa. Han e Na (1994) desenvolveram um abordagem de dois estágios, que inicialmente posiciona itens observando a placa e seus defeitos para, em seguida, melhorar a ocupação da placa com uma heurística baseada no recozimento simulado. Outros trabalhos envolvendo placas irregulares foram feitos em Tay, Chong e Lee (2002), que desenvolveram um algoritmo genético, e Yuping, Shouwei e Chunli (2005), que consideraram zonas de qualidades para a placa e uma estratégia de penalização para achar posições válidas. Crispin *et al.* (2005) desenvolveram algoritmos genéticos para um problema na indústria de calçados. Os algoritmos usam o *no-fit polygon* para obter posições que

maximizam a utilização da placa ou que mantêm os itens em contato. Em Lee, Ma e Cheng (2008), o problema considera múltiplas placas irregulares e com defeitos, sendo que os itens são ordenados pelo valor e empacotados inicialmente nesta ordem. Visando melhorar a solução, o item pode ser rotacionado e/ou transladado da sua posição inicial.

A resolução de um problema de corte de peças de couro para a indústria automobilística foi feito em Alves *et al.* (2012a) e Alves *et al.* (2012b). Em Alves *et al.* (2012b), buscou-se pela maximização da ocupação de uma única placa, sendo desenvolvida uma heurística de busca em vizinhança variável que considera a otimização sobre a sequência de itens. Os autores desenvolveram quatro operadores para mover itens dentro da sequência em busca de uma sequência que resulte em uma solução melhor. Mundim e Andretta (2014) consideraram o problema com vários recipientes irregulares, visando minimizar o número de recipientes necessários para o corte dos itens demandados. Os autores apresentaram duas heurísticas baseadas na regra de posicionamento *bottom-left*. A não sobreposição entre os itens e com os defeitos é garantida pelo *no-fit polygon*. Em Pinto *et al.* (2016) há uma melhora sobre os resultados de Alves *et al.* (2012b), com o uso de uma heurística construtiva que simula o posicionamento de itens em pontos da placa, selecionando aquele item e ponto que gerem a melhor ocupação da placa. Para melhorar a solução, os autores propuseram uma busca local, em que itens já posicionados são removidos de suas posições e itens ainda não posicionados são testados nessas posições para ver se gera uma melhor ocupação da placa.

Reijntjes (2016) considerou a placa com defeitos e regiões de qualidade, propondo heurísticas construtivas baseadas na *bottom-left* e consideradas dentro de um algoritmo *branch-and-bound*. A não sobreposição entre itens é garantida pelo cálculo da região livre de colisão, que se baseia no *no-fit polygon*. Chen *et al.* (2020) lidaram com o problema da mochila em que os itens são retangulares e a placa é irregular e com defeitos, com aplicação no corte de ardósias. O objetivo é maximizar a ocupação da placa considerando que os itens são obtidos a partir de cortes guilhotinados. A heurística dos autores considera a placa subdividida por níveis horizontais e os itens são posicionados nas sub-placas. Sequências de itens são geradas e controladas por um algoritmo genético.

A literatura é ainda mais escassa quando se busca pelo tratamento de incertezas dentro de problemas de corte de itens irregulares. Mundim (2017) considerou incertezas na demanda dos itens para o problema com múltiplas placas, propondo um modelo de programação estocástica de dois estágios. No modelo se penaliza a falta ou o excesso de itens produzidos a mais que o necessário, conforme um conjunto amostrado de cenários. Diferentemente desses autores, lida-se com o problema da mochila com itens irregulares considerando que os defeitos na placa são dados incertos. Propõe-se um modelo de programação estocástica de dois estágios com recurso. O primeiro estágio considera decisões sobre quais itens selecionar para obter da placa, enquanto o segundo estágio contém os cenários com a realização dos defeitos na placa e, então, alguns itens poderão ser cancelados por não mais haver um posicionamento viável na placa. Para obter

um plano de corte viável em cada cenário, considera-se que o posicionamento dos itens é feito observando o *no-fit polygon* discretizado sobre uma matriz de pontos, sendo denominado *no-fit raster* (TOLEDO *et al.*, 2013).

O modelo de programação estocástica proposto assume uma aplicação de longo prazo (isto é, é de risco neutro). Objetivando controlar a variabilidade que se possa ocorrer pelas realizações dos defeitos, assim buscando uma solução robusta, avalia-se também uma medida de risco para controlar a variabilidade das decisões de segundo estágio (AHMED; SAHINIDIS, 1998), resultando em um modelo de aversão ao risco. Realizam-se experimentos computacionais com 40 instâncias adaptadas da literatura de problemas de corte de itens irregulares. Para cada instância, cenários são criados a partir de uma árvore de possibilidades para a ocorrência de cada defeito e quatro casos de probabilidade para cada cenário são estudados. Os experimentos computacionais avaliam o valor esperado da informação perfeita, o valor da solução estocástica e a medida de aversão ao risco, concluindo sobre a necessidade de se considerar as incertezas a fim de obter soluções economicamente viáveis para o problema.

Este trabalho está organizado da seguinte maneira. A definição do problema é dada na Seção 4.2, enquanto a Seção 4.3 apresenta o modelo proposto, discute as análises para avaliar o impacto das incertezas no problema e introduz o modelo averso ao risco. A Seção 4.4 contém o estudo computacional realizado sobre o modelo, que é resolvido por um algoritmo *branch-and-cut* e considera a resolução de instâncias adaptadas da literatura de problemas de corte de itens irregulares. Por fim, considerações finais são dadas na Seção 4.5.

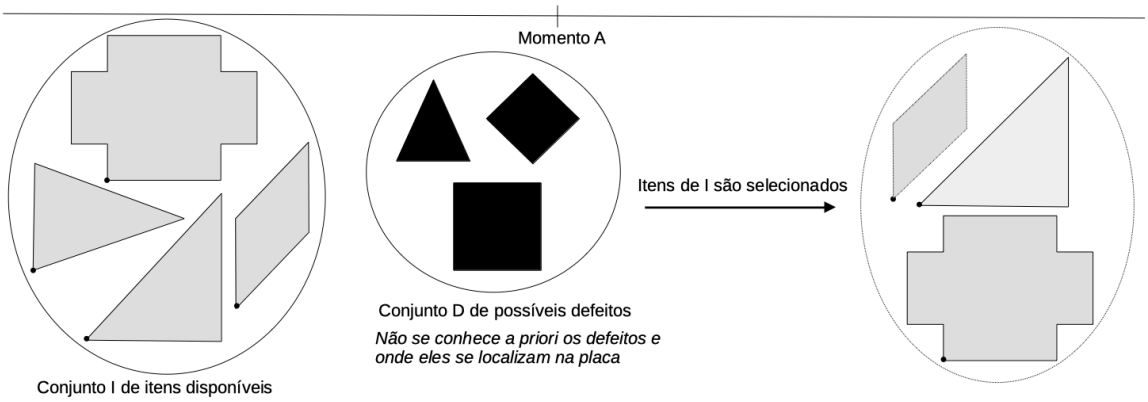
4.2 Definição do problema

O problema da mochila bidimensional com itens irregulares (isto é, 2IKP) é uma generalização do problema da mochila bidimensional, que é NP-Difícil (GAREY; JOHNSON, 1979). O problema em estudo é definido sobre o plano Cartesiano, com uma placa posicionada no primeiro quadrante e com origem em $(0,0)$. O eixo das abscissas (eixo x) está associado com a dimensão da largura L e o eixo das ordenadas (eixo y) está associado com a dimensão da altura/comprimento H, que são valores fixos e conhecidos *a priori*. Existe um conjunto de itens I disponíveis para seleção. Cada item $i \in I$ possui área a_i e uma envoltória retangular, que é o menor retângulo, em termos de lados e sem rotação, que circunscreve o item. O item não pode ser rotacionado e é posicionado pelo seu vértice de referência, que é o vértice de menor ordenada e, em caso de empate, o de menor abscissa. Além disso, cada item i tem lucro v_i, dada a sua seleção e produção, e custo de cancelamento c_i, dada a sua seleção e não produção.

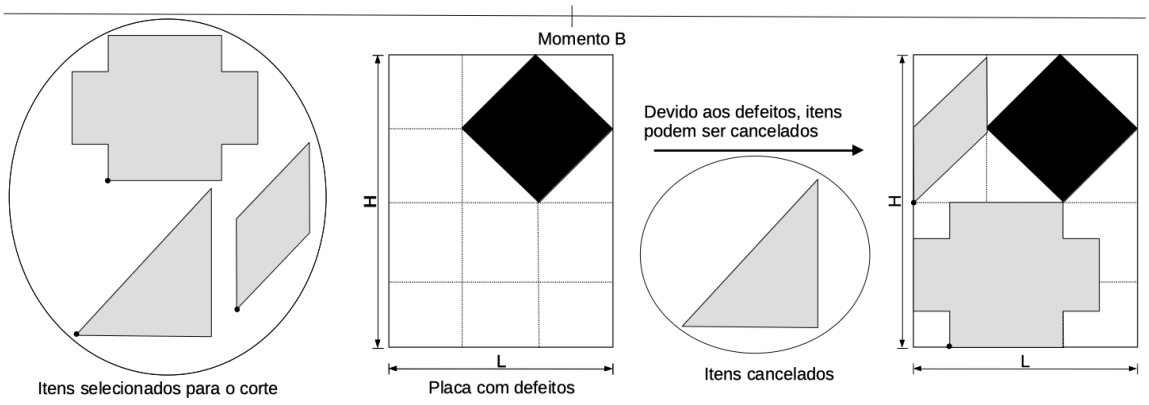
O 2IKP-DU considera um conjunto D de possíveis defeitos que a placa pode ter. Cada defeito $d \in D$ é modelado como um item regular ou irregular de dimensões, área e vértice de referência conhecidos. Por outro lado, não se conhece quais defeitos e seu posicionamento dentro da placa, sendo tratados como dados incertos. O objetivo do problema é determinar um plano de

corte viável que retorne o máximo lucro ao cortar a placa. Um plano de corte é viável quando os itens a serem produzidos podem ser posicionados sem sobreposição, inteiramente dentro da placa e não contêm partes com defeito.

A seleção dos itens que poderão fazer parte do plano de corte ocorre em um momento inicial A, em que se faz o pedido da placa. Nesse momento, selecionam-se itens do conjunto I que deveriam ser produzidos a partir do corte da placa retangular de dimensões conhecidas. Como a placa pode apresentar defeitos que são conhecidos somente no momento B (de entrega e produção dos itens), alguns itens selecionados poderão ser cancelados, incorrendo um custo de cancelamento pela não produção. O plano de corte somente é confirmado no segundo momento, em que há o conhecimento dos defeitos e, assim, pode-se confirmar quais dos itens selecionados se pode posicionar de forma viável, enquanto alguns poderão vir a ser cancelados. Dessa forma, deseja-se um plano de corte viável que resulte no maior lucro possível. A Figura 4.1 ilustra um exemplo do 2IKP-DU e as decisões que precisam ser tomadas em cada momento.



(a) Seleção de itens no momento A



(b) Confirmação do plano de corte no momento B

Figura 4.1 – Decisões que precisam ser tomadas para obter uma solução para o 2IKP-DU.

Com relação ao posicionamento de itens na placa, usa-se o conceito de *no-fit raster* para garantir a não sobreposição entre itens e o conceito de *inner-fit raster* para assegurar que os itens estejam completamente dentro da placa (TOLEDO *et al.*, 2013). Para o *no-fit raster*, obtém-se primeiramente o *no-fit polygon* NFP_{ij} para cada par de itens i e j, sendo um item fixo (i) e o

outro orbital (j). O item orbital é transladado ao redor do item fixo, sempre encostando no item fixo, mas sem gerar sobreposição, formando um polígono cujo interior indica sobreposição entre os itens. Esse polígono é, então, discretizado em uma matriz de pontos, com valores iguais a "um" indicando a sobreposição; caso contrário, com valores iguais a "zero". Para o *inner-fit raster*, obtém-se inicialmente o *inner-fit polygon* IFP_i, que consiste no polígono obtido ao transladar cada item i, internamente, sempre encostando em uma das bordas da placa. Esse polígono é, então, discretizado em uma matriz de pontos, com valores iguais a "zero" indicando que o item pode ser posicionado sem extrapolar as dimensões da placa; caso contrário, com valores iguais a "um".

4.3 Modelo de programação estocástica

Desenvolve-se um modelo de programação estocástica de dois estágios com recurso para o 2IKP-DU. Os defeitos (e suas localizações na placa) são considerados variáveis aleatórias com realizações discretas conforme uma distribuição de probabilidade conhecida. Seja $\Omega = \{1,2,\ldots,S\}$ o conjunto de possíveis estados, isto é, de cenários $s \in \Omega$ amostrados, com a realização dos defeitos na placa.

No modelo proposto, o primeiro estágio do problema está relacionado com decisões "aqui-e-agora" (*here-and-now*), envolvendo a seleção de itens para serem produzidos a partir do corte da placa. Por outro lado, no segundo estágio, decisões "espere-e-veja" (*wait-and-see*) são tomadas para o posicionamento dos itens selecionados. Devido à realização das variáveis aleatórias neste estágio, isto é, o conhecimento dos defeitos na placa, existem ainda decisões envolvendo o possível cancelamento de itens. O plano de corte da placa deve ser viável e ter o máximo lucro esperado, que envolve o lucro com os itens selecionados e produzidos menos o custo esperado com os itens cancelados (isto é, selecionados, mas não produzidos). Os parâmetros e as variáveis de decisão do modelo de programação estocástica são:

- I: conjunto de itens irregulares disponíveis para a seleção;
- v_i: lucro associado com a seleção do item i no primeiro estágio;
- c_i: custo associado ao cancelamento do item i no segundo estágio;
- π^s: probabilidade de ocorrência do cenário $s \in \Omega$;
- D^s: realização dos defeitos e suas localizações na placa, dado o cenário $s \in \Omega$;
- M: número suficientemente grande.
- y_i: variável binária, de primeiro estágio, que recebe o valor 1 se o item $i \in I$ faz parte dos itens selecionados para a produção; caso contrário, a variável recebe o valor 0;

- x^s_{ipq}: variável binária, de segundo estágio, que recebe o valor 1 se o item i tem o seu vértice de referência posicionado no ponto (p,q) da malha associada a placa; caso contrário, a variável recebe o valor 0; para cada cenário $s \in \Omega$;

- z^s_i: variável binária, de segundo estágio, que recebe o valor 0 se o item i deve ser cancelado; caso contrário, a variável recebe o valor 1; para cada cenário $s \in \Omega$;

O modelo do 2IKP-DU é de programação estocástica com dois estágios e risco neutro, assumindo que a solução terá desempenho de longo prazo. O objetivo é maximizar o lucro líquido, considerando a seleção dos itens a produzir (primeiro estágio) e o posicionamento e possível cancelamento de itens conforme os defeitos na placa (segundo estágio). A função objetivo (4.1) do primeiro estágio considera o lucro pela seleção de itens ($\sum_{i\in I} v_i y_i$) e o custo esperado pelo cancelamento de itens dado pela função de recurso $Q(\boldsymbol{y}, \boldsymbol{\xi})$, em que $\boldsymbol{\xi} = [\xi_s]$, com $\xi_s = \{x^s_{ipq}, z^s_i\}$ representando a realização das variáveis aleatórias através do cenário s. A restrição (4.2) impõe que os itens selecionados respeitem a área total do recipiente, enquanto as restrições (4.3) definem o domínio das variáveis de decisão de primeiro estágio.

$$\text{Maximizar} \sum_{i\in I} v_i y_i + Q(\boldsymbol{y}, \boldsymbol{\xi}) \tag{4.1}$$

$$\text{sujeito a:} \sum_{i\in I} a_i y_i \leq LH \tag{4.2}$$

$$y_i \in \{0,1\}, \qquad \forall\, i \in I. \tag{4.3}$$

O modelo de segundo estágio possui função objetivo (4.4), que busca pela minimização da soma dos custos esperados de cancelamento de itens considerando a probabilidade de ocorrência π^s de cada cenário $s \in \Omega$. Como itens selecionados, porém cancelados, não podem ter seu lucro contabilizado na solução, assume-se que $c_i \geq v_i$, para cada item $i \in I$.

$$Q(\boldsymbol{y}, \boldsymbol{\xi}) = \text{Minimizar} \sum_{s\in\Omega} \pi_s \left(\sum_{i\in I} c_i (y_i - z^s_i) \right) \tag{4.4}$$

$$\text{sujeito a:} \sum_{j\in I} \sum_{(u,v)\in NFP^{(p,q)}_{ij}} x^s_{juv} \leq (1 - x^s_{ipq})M, \qquad \forall\, s \in \Omega,\ i \in I,\ (p,q) \in IFP^s_i; \tag{4.5}$$

$$z^s_i \leq y_i, \qquad \forall\, s \in \Omega,\ i \in I; \tag{4.6}$$

$$\sum_{(p,q)\in IFP^s_i} x^s_{ipq} = z^s_i, \qquad \forall\, s \in \Omega,\ i \in I; \tag{4.7}$$

$$z^s_i \in \{0,1\}, \qquad \forall\, s \in \Omega,\ i \in I; \tag{4.8}$$

$$x^s_{ipq} \in \{0,1\}, \qquad \forall\, s \in \Omega,\ i \in I,\ (p,q) \in IFP^s_i. \tag{4.9}$$

As restrições (4.5) asseguram, para cada cenário $s \in \Omega$, que se o item i está posicionado no ponto (p,q) da placa, então o item j não pode ser posicionado em qualquer um dos pontos (u,v) do conjunto $NFP_{ij}^{(p,q)}$. As restrições (4.6) do modelo de segundo estágio impõem que um item não selecionado no primeiro estágio deve ser marcado com o mesmo estado que os itens cancelados, dado cada cenário $s \in \Omega$. As restrições (4.7) impõem que somente os itens i selecionados e não cancelados devem ser posicionados na placa, para cada cenário $s \in \Omega$. Por fim, as restrições (4.8) e (4.9) definem que as variáveis de segundo estágio são binárias. Para ajudar na não sobreposição de itens, pode-se usar restrições adicionais que impõem que cada ponto da malha seja coberto por no máximo um item (JUNQUEIRA, 2009).

O conjunto $NFP_{ij}^{(p,q)}$ contém os pontos (u,v) que estão no interior do NFP_{ij}, entre i, o item fixo, e j, o item orbital, dado que i está com o seu vértice de referência posicionado em (p,q). Os pontos (u,v) são aqueles em que, se o item j tem o seu vértice de referência posicionado sobre algum deles, faz com que haja sobreposição com i que está posicionado em (p,q). Por outro lado, o conjunto $IFP_i^s \subseteq IFP_i$ contém os pontos da placa, dado o cenário $s \in \Omega$, onde o vértice de referência do item i pode ser posicionado de forma que o item não contenha partes com defeito e que esteja inteiramente dentro na placa.

4.3.1 Análise da solução do modelo estocástico

Uma solução para um problema determinístico contém um único plano para o corte da placa, enquanto na solução do modelo estocástico de dois estágios com recurso há $|\Omega|$ planos de corte. Dessa forma, avaliam-se as soluções para o 2IKP-DU através do cálculo do valor esperado da informação perfeita e o valor da solução estocástica. Essas análises permitem obter conclusões sobre a importância em considerar (ou não) as incertezas do problema. A solução do modelo de programação estocástica (de risco neutro) proposto para o 2IKP-DU é obtida da resolução do seu modelo equivalente determinístico (de risco neutro), por qualquer *solver* de programação linear inteira. Esse modelo consiste na função objetivo (4.1), com a função de recurso $Q(\boldsymbol{y}, \boldsymbol{\xi})$ dada pela função (4.4), e as restrições (4.2), (4.3) e (4.5)-(4.9).

O Valor Esperado da Informação Perfeita (*Expected Value of Perfect Information - EVPI*) indica até quanto um decisor estaria disposto a pagar para ter informação completa e precisa sobre o futuro. No caso do 2IKP-DU, essas informações estão relacionadas ao conhecimento exato dos futuros defeitos (e suas localizações na placa), possibilitando obter um plano para o corte da placa que resulte no máximo lucro possível.

Para calcular o EVPI de uma instância do problema: (i) obtém-se a solução do modelo equivalente determinístico (*Recourse Problem - RP*) com todos os cenários em Ω, resultando na solução RP; (ii) obtém-se a solução do problema *wait-and-see* para cada cenário individualmente, ou seja, resolve-se o modelo equivalente determinístico para cada cenário $s \in \Omega$, resultando na solução WS_s^*; (iii) calcula-se o valor esperado WS obtido das soluções dos problemas *wait-and-see*, isto é, $\text{WS} = \sum_{s\in\Omega} \pi^s \text{WS}_s^*$; (iv) calcula-se EVPI como (WS - RP). Um valor pequeno do

EVPI indica que as incertezas do problema não são tão impactantes na solução, não valendo a pena resolver um modelo de programação estocástica, sendo mais oportuno (fácil) usar uma solução aproximada.

Enquanto a obtenção do EVPI pode requerer a resolução de vários problemas, uma outra alternativa para avaliar soluções do 2IKP-DU é o cálculo do Valor da Solução Estocástica (*Value of the Stochastic Solution - VSS*). O VSS indica o custo de ignorar as incertezas pelo uso da solução de um problema de valor esperado. No caso do 2IKP-DU, isso corresponderia a tomar decisões com base em um cenário de referência cujos valores das variáveis aleatórias são conhecidos, ou seja, assumindo uma dada realização dos defeitos (e suas localizações na placa).

Para obter o VSS de uma instância do problema: (i) obtém-se a solução RP do modelo equivalente determinístico com todos os cenários em Ω; (ii) obtém-se a solução de um problema de Valor Esperado (*Expected Value problem - EV*). Neste caso, assume-se um cenário de referência, que dentre os cenários disponíveis para a instância, é o pior em termos da quantidade de defeitos na placa. A solução do EV é obtida da resolução do modelo equivalente determinístico para esse dado cenário de referência, resultando nos valores $\bar{\mathbf{y}}$ para as variáveis y_i; (iii) calcula-se o Resultado Esperado do problema de Valor Esperado (*Expectation of the Expected Value Problem - EVV*), ou seja, a solução do EVV é obtida da resolução do modelo equivalente determinístico com todos os cenários em Ω, porém com as variáveis y_i (de primeiro estágio) fixas nos valores em $\bar{\mathbf{y}}$. (iv) calcula-se VSS como (RP - EVV). Um valor pequeno de VSS indica que o ganho da solução do modelo de programação estocástica (RP) sobre a solução de um problema de valor esperado (EV) é pequeno, sendo mais oportuno (fácil) obter uma solução aproximada.

4.3.2 Aversão ao risco

Em um modelo neutro ao risco, a solução tem desempenho de longo prazo, de forma que os cenários representam tendências gerais para as variáveis aleatórias, ou seja, não se espera uma dispersão das variáveis aleatórias que impacte significativamente na solução obtida. Quando existe variabilidade nos parâmetros incertos, por sua vez, ocasionando dispersão das variáveis aleatórias, o valor esperado do custo, que representa a função objetivo de segundo estágio, pode ter alta variabilidade. Em situações de risco, havendo alta variabilidade dos parâmetros, o valor esperado do custo de segundo estágio deveria ser controlado (limitado). Uma forma de controlar o risco é considerar um modelo de programação estocástica averso ao risco, de forma que o custo esperado de cada cenário esteja próximo do custo esperado ótimo quando se considera todos os cenários (ALEM; MORABITO, 2015a).

Uma medida de risco para limitar a variabilidade das variáveis de segundo segundo estágio, consequentemente, obter soluções menos sensíveis a variabilidade dos cenários (isto é, mais robustas e, assim, aversas ao risco) é minimizar a variância do valor esperado total, porém isso introduz não linearidades no modelo (BIRGE; LOUVEAUX, 1997). Por outro lado, (AHMED; SAHINIDIS, 1998) propuseram uma medida de risco, denominada Média Parcial

Superior (*Upper Partial Mean - UPM*), que pode ser usada como uma "medida de variância" definida sobre funções lineares da forma:

$$\sum_{s\in\Omega} \pi^s \Delta^s \tag{4.10}$$

sendo Δ^s a diferença entre o valor esperado do custo de cada cenário e o valor total esperado do custo de todos os cenários.

Uma forma de inserir a medida (4.10) é considerar o modelo de programação estocástica de dois estágios com recurso restrito. Neste caso, reduz-se a variabilidade da solução de segundo estágio (isto é, do valor esperado do custo de cancelamento) impondo uma tolerância máxima $\Delta^{\max}$. Dessa forma, o modelo equivalente determinístico averso ao risco para o 2IKP-DU é definido pela função objetivo (4.11) e as restrições (4.2), (4.3), (4.6)-(4.9), (4.12)-(4.14):

$$\text{Maximizar } \sum_{i\in I} v_i y_i - \sum_{s\in\Omega} \pi^s \left(\sum_{i\in I} c_i (y_i - z_i^s) \right) \tag{4.11}$$

sujeito a: (4.2), (4.3), (4.5), (4.6), (4.7), (4.8), (4.9),

$$\Delta^s \geq \sum_{i\in I} c_i (y_i - z_i^s) - \left(\sum_{\bar{s}\in\Omega} \pi^{\bar{s}} \left(\sum_{i\in I} c_i (y_i - z_i^{\bar{s}}) \right) \right), \qquad \forall\, s \in \Omega; \tag{4.12}$$

$$\sum_{s\in\Omega} \pi^s \Delta^s \leq \alpha \Delta^{\max}; \tag{4.13}$$

$$\Delta^s \geq 0, \qquad \forall\, s \in \Omega. \tag{4.14}$$

Na restrição (4.13), quando $\alpha\Delta^{\max}$ for pequeno, espera-se que a solução resultante seja menos sensível às variações dos cenários e, assim, o decisor é mais conservador (averso ao risco). Valores muitos pequenos dessa tolerância podem tornar o modelo infactível, ao passo que valores muito grandes podem resultar na solução do modelo de programação estocástica de risco neutro. A ideia é construir uma curva de soluções variando o parâmetro $\alpha \in [0,1]$. Assume-se que $\Delta^{\max} = \sum_{s\in\Omega} \pi^s \Delta^s$ é obtido da solução RP do modelo equivalente determinístico de risco neutro com todos os cenários.

4.4 Resultados computacionais

Os modelos equivalentes determinísticos de risco neutro (4.1)-(4.9) e averso ao risco (4.11)-(4.14) foram implementados, na linguagem C++, dentro do *framework* do algoritmo *branch-and-cut* do Gurobi (GUROBI OPTIMIZATION, 2020), versão 9.1, usando as configurações padrões. O computador usado nos experimentos numéricos possui sistema operacional Linux Ubuntu 16.04 LTS, processador Intel Xeon E3-1245v5 de 3,5 GHz e 32 GB de RAM. A

resolução das instâncias pelo Gurobi considera um tempo limite de 7200 segundos (2 horas) por instância.

4.4.1 Instâncias

As 40 instâncias utilizadas nos testes foram adaptadas da literatura do problema da mochila bidimensional com itens irregulares e disponibilizadas por Souza Queiroz e Andretta (2020b). Algumas dessas instâncias podem ser encontradas no *EURO Special Interest Group on Cutting and Packing - ESICUP*[1]). Os principais dados de cada instância são apresentados na Tabela 4.1, contendo o nome, a quantidade total de itens disponíveis para a seleção e as dimensões da placa. Adota-se a escala de 1 ponto a cada 1 unidade de distância para obter as matrizes de *no-fit raster* e *inner-fit raster*. Com relação à função objetivo de primeiro e segundo estágios, considera-se $v_i = a_i$, isto é, o lucro corresponde ao valor da área do item, e $c_i = 1,5v_i$, para cada item $i \in I$.

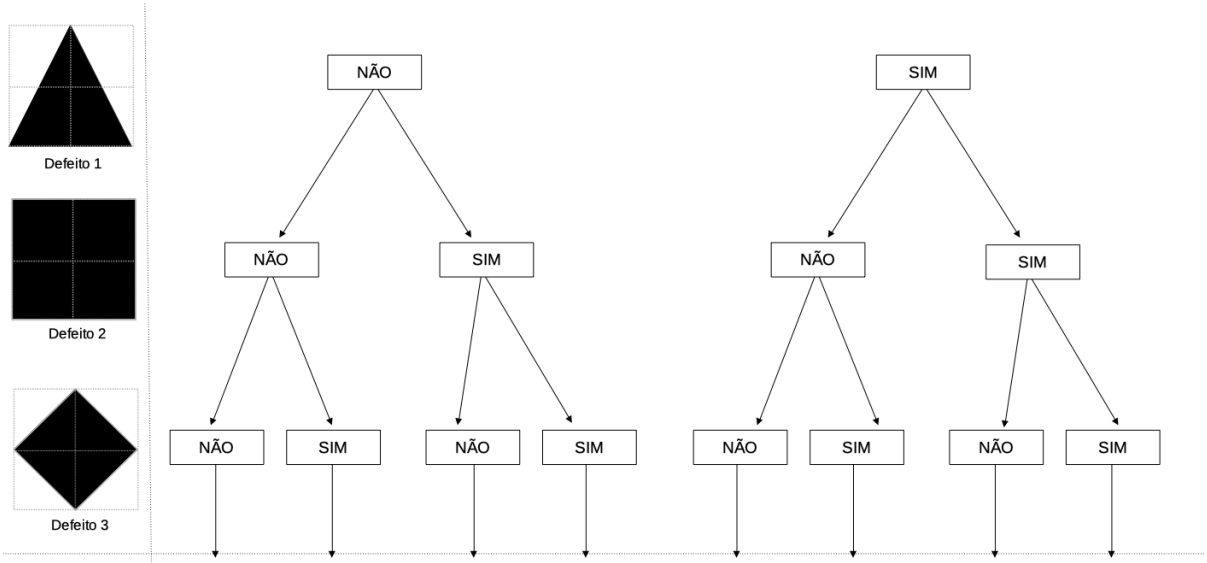
Tabela 4.1 – Instâncias adaptadas para os testes computacionais com o 2IKP-DU.

Instância	Trabalho	#Itens	H	L	Instância	Trabalho	#Itens	H	L
blasz2	Oliveira, Gomes e Ferreira (2000)	16	15	16	rco2	Toledo *et al.* (2013)	14	15	13
blazewicz1	Toledo *et al.* (2013)	7	15	6	rco3	Toledo *et al.* (2013)	21	15	19
blazewicz2	Toledo *et al.* (2013)	14	15	11	rco4	Toledo *et al.* (2013)	28	15	26
blazewicz3	Toledo *et al.* (2013)	21	15	17	rco5	Toledo *et al.* (2013)	35	15	32
blazewicz4	Toledo *et al.* (2013)	28	15	22	shapes2	Toledo *et al.* (2013)	8	40	14
blazewicz5	Toledo *et al.* (2013)	35	15	27	shapes4	Toledo *et al.* (2013)	16	40	16
dagli1	Rodrigues e Toledo (2017)	10	60	23	shapes5	Toledo *et al.* (2013)	20	40	20
fu	Fujita, Akagi e Hirokawa (1993)	12	38	29	shapes7	Toledo *et al.* (2013)	28	40	28
fu5	Alvarez-Valdes, Martinez e Tamarit (2013)	5	38	14	shapes9	Toledo *et al.* (2013)	34	40	33
fu6	Alvarez-Valdes, Martinez e Tamarit (2013)	6	38	17	shapes15	Toledo *et al.* (2013)	43	40	40
fu7	Alvarez-Valdes, Martinez e Tamarit (2013)	7	38	19	shirts1-2	Rodrigues e Toledo (2017)	13	40	13
fu8	Alvarez-Valdes, Martinez e Tamarit (2013)	8	38	20	shirts2-4	Rodrigues e Toledo (2017)	26	40	14
fu9	Alvarez-Valdes, Martinez e Tamarit (2013)	9	38	23	shirts3-6	Rodrigues e Toledo (2017)	39	40	21
fu10	Alvarez-Valdes, Martinez e Tamarit (2013)	10	38	26	shirts4-8	Rodrigues e Toledo (2017)	52	40	28
poly1a	Hopper (2000)	15	40	13	shirts5-10	Rodrigues e Toledo (2017)	65	40	35
poly1b	Rodrigues e Toledo (2017)	15	40	13	three	Alvarez-Valdes, Martinez e Tamarit (2013)	3	7	4
poly1c	Rodrigues e Toledo (2017)	15	40	13	threep2	Alvarez-Valdes, Martinez e Tamarit (2013)	6	7	7
poly1d	Rodrigues e Toledo (2017)	15	40	11	threep2w9	Alvarez-Valdes, Martinez e Tamarit (2013)	6	9	6
poly1e	Rodrigues e Toledo (2017)	15	40	10	threep3	Alvarez-Valdes, Martinez e Tamarit (2013)	9	7	10
rco1	Toledo *et al.* (2013)	7	15	7	threep3w9	Alvarez-Valdes, Martinez e Tamarit (2013)	9	9	8

A geração dos cenários considera uma árvore de possibilidades (MA; KWON; LEE, 2010), com três tipos diferentes de defeitos, representados por um triângulo, um quadrado e um losango. Cada cenário considera a presença ou não de cada um dos defeitos e a posição do defeito na placa foi definida de forma aleatória usando uma distribuição uniforme. Há um total de 8 cenários por instância, que são as possibilidades entre não ter qualquer defeito na placa e ter todos os defeitos na placa. Cada instância está associada a quatro casos de probabilidade para a ocorrência dos cenários (ALEM; MORABITO, 2015a), que são: *equiprovável*, com a probabilidade de 50% associada a ter o defeito d na placa e os outros 50% para não ter tal defeito presente na placa; *moderado*, com a probabilidade de 60% associada a ter o defeito d na placa e os outros 40% para não ter tal defeito presente na placa; *otimista*, com a probabilidade de 25% associada a ter o defeito d na placa e os outros 75% para não ter tal defeito presente na placa; e *pessimista* com a probabilidade de 75% associada a ter o defeito d na placa e

[1] https://www.euro-online.org/websites/esicup/data-sets

os outros 25% para não ter tal defeito presente na placa. Por exemplo, para o caso *otimista*, o cenário onde não há qualquer defeito na placa possui probabilidade de ocorrência igual a $75\% \times 75\% \times 75\% = 42,19\%$, em que a probabilidade de 75% está associada a *não* ter o defeito d na placa. Por outro lado, no cenário cuja placa contém todos os defeitos, a sua probabilidade de ocorrência é dada por $25\% \times 25\% \times 25\% = 1,56\%$, com a probabilidade de 25% associada a *sim*, ou seja, de ter o defeito d na placa, para $d = 1,2,3$. A Figura 4.2 apresenta a árvore de cenários com a probabilidade de ocorrência de cada cenário para cada um dos casos.



Cenário	#1. NÃO-NÃO-NÃO	#2. NÃO-NÃO-SIM	#3. NÃO-SIM-NÃO	#4. NÃO-SIM-SIM	#5. SIM-NÃO-NÃO	#6. SIM-NÃO-SIM	#7. SIM-SIM-NÃO	#8. SIM-SIM-SIM
Probabilidades								
Moderado:	6,40%	9,60%	9,60%	14,40%	9,60%	14,40%	14,40%	21,60%
Equiprovável:	12,50%	12,50%	12,50%	12,50%	12,50%	12,50%	12,50%	12,50%
Otimista:	42,19%	14,06%	14,06%	4,69%	14,06%	4,69%	4,69%	1,56%
Pessimista:	1,56%	4,69%	4,69%	14,06%	4,69%	14,06%	14,06%	42,19%

Figura 4.2 – Árvore de cenário para as instâncias usadas nos testes computacionais.

4.4.2 Resultados do modelo de risco neutro

As Tabelas 4.2 a 4.5 contêm os resultados obtidos ao resolver o modelo equivalente determinístico de risco neutro (4.1)-(4.9) sobre as instâncias da Tabela 4.1, considerando os 8 cenários e quatros casos (moderado, equiprovável, otimista e pessimista) apresentados na Figura 4.2. Essas tabelas contêm informações sobre o nome da instância, o total de itens disponíveis para a seleção, o número de itens selecionados para a possível produção, o valor da função objetivo, o *gap* retornado pelo Gurobi, o tempo computacional requerido pelo Gurobi e a quantidade de itens cancelados dado cada um dos 8 cenários. De maneira geral é possível obter a solução ótima para 42,5% das instâncias dentro do tempo limite de 7200 segundos, com um *gap* médio de 5,0% e um tempo computacional médio de 4284,1 segundos. As instâncias não resolvidas na otimalidade tendem a ter um maior quantitativo de itens para a seleção ou uma placa de dimensões maiores.

Analisando os resultados para cada um dos casos individualmente, para o caso moderado, na Tabela 4.2, o *gap* e o tempo médio são de 5,0% e 4185,9 segundos, respectivamente. Nesse caso, soluções ótimas são obtidas para 45,0% das instâncias, enquanto a instância com maior *gap* é a shapes15, com um valor de 24,3%. Não há qualquer cancelamento de itens para 62,5% instâncias, incluindo as instâncias poly1c, shapes2 e shirts1-2, que também têm todos os itens selecionados e produzidos. O maior número de cancelamentos ocorre nos cenários #4, #6, #7 e #8, enquanto as instâncias rco1 e threep2w9 possuem mais cancelamentos de itens.

Tabela 4.2 – Resultados do caso moderado para o 2IKP-DU.

Instância	Total de itens	#Itens selecionados	Obj.	*gap* (%)	Tempo (s)	#Itens cancelados no cenário							
						#1	#2	#3	#4	#5	#6	#7	#8
blasz2	16	12	164,4	0,0	35,3	0	0	0	0	0	1	1	0
blazewicz1	7	5	46,9	0,0	1,5	0	0	0	1	0	0	1	1
blazewicz2	14	11	113,3	0,0	220,7	0	0	0	0	0	1	0	0
blazewicz3	21	18	198,9	1,6	7200,0	0	0	0	0	0	0	0	1
blazewicz4	28	23	263,2	3,4	7200,0	0	0	0	0	0	0	0	0
blazewicz5	35	28	320,5	7,4	7200,0	0	0	0	0	0	0	0	0
dagli1	30	14	1094,3	18,4	7200,0	1	0	0	0	0	1	1	0
fu	12	9	887,0	8,2	7200,0	0	0	0	0	0	0	0	0
fu5	5	4	314,4	0,0	2,2	0	0	0	0	0	1	1	0
fu6	6	4	434,0	0,0	6,5	0	0	0	0	0	0	0	0
fu7	7	4	441,0	0,0	272,6	0	0	0	0	0	0	0	0
fu8	8	6	537,3	0,0	3650,5	0	0	0	0	0	0	0	1
fu9	9	8	651,0	0,0	4336,4	0	0	0	0	0	0	0	0
fu10	10	8	790,1	5,6	7200,0	0	0	0	0	0	0	0	1
poly1a	15	11	310,8	18,7	7200,0	0	0	0	1	0	0	1	1
poly1b	15	10	313,0	20,6	7200,0	0	0	0	0	0	0	0	0
poly1c	15	15	315,5	0,0	328,7	0	0	0	0	0	0	0	0
poly1d	15	13	261,5	22,5	7200,0	0	0	0	0	0	0	0	0
poly1e	15	12	241,5	4,8	7200,0	0	0	0	0	0	0	0	0
rco1	7	6	57,0	0,0	10,8	0	0	0	1	0	1	1	2
rco2	14	12	157,0	0,0	126,8	0	0	0	0	0	0	0	0
rco3	21	19	236,5	1,3	7200,0	0	0	0	0	0	0	0	0
rco4	28	24	333,0	3,8	7200,0	0	0	0	0	0	0	0	0
rco5	35	30	411,3	7,8	7200,0	0	0	0	0	0	0	0	0
shapes2	8	8	315,7	0,0	36,3	0	0	0	0	0	0	0	0
shapes4	16	11	375,9	0,3	7200,0	0	0	0	0	0	0	0	0
shapes5	20	13	510,6	3,0	7200,0	0	0	0	0	0	0	0	0
shapes7	28	18	780,0	7,2	7200,0	0	0	0	0	0	0	0	0
shapes9	34	23	903,3	12,1	7200,0	0	0	0	0	0	0	0	0
shapes15	43	27	1032,0	24,3	7200,0	0	0	0	0	0	0	0	0
shirts1-2	13	13	275,0	0,0	6,1	0	0	0	0	0	0	0	0
shirts2-4	26	22	439,2	0,2	7200,0	0	0	0	0	0	0	0	0
shirts3-6	39	37	714,3	1,7	7200,0	0	0	0	0	0	0	0	0
shirts4-8	52	48	948,9	4,9	7200,0	0	0	0	0	0	0	0	0
shirts5-10	65	62	1117,0	23,1	7200,0	0	0	0	0	0	0	0	0
three	3	1	2,2	0,0	0,0	0	0	0	1	0	0	1	1
threep2	6	3	17,8	0,0	0,1	0	0	0	0	0	1	0	1
threep2w9	6	4	20,8	0,0	0,1	0	0	0	1	1	0	1	2
threep3	9	5	35,2	0,0	0,2	0	0	0	0	0	1	0	2
threep3w9	9	5	37,1	0,0	0,4	0	0	0	0	0	0	1	1
Média	-	-	-	5,0	4185,9	-	-	-	-	-	-	-	-
Des. Padrão	-	-	-	7,6	3478,9	-	-	-	-	-	-	-	-

Tabela 4.3 – Resultados do caso equiprovável para o 2IKP-DU.

Instância	Total de itens	#Itens selecionados	Obj.	*gap* (%)	Tempo (s)	#Itens cancelados no cenário							
						#1	#2	#3	#4	#5	#6	#7	#8
blasz2	16	12	167,6	0,0	68,9	0	0	0	0	0	1	1	0
blazewicz1	7	5	49,1	0,0	2,1	0	0	0	1	0	0	1	1
blazewicz2	14	11	117,0	0,0	510,1	0	0	0	0	0	1	0	0
blazewicz3	21	18	201,6	1,6	7200,0	0	0	0	0	0	0	0	1
blazewicz4	28	23	263,6	4,2	7200,0	0	0	0	0	0	0	0	0
blazewicz5	35	28	328,5	5,2	7200,0	0	0	0	0	0	0	0	0
dagli1	30	14	1115,3	19,1	7200,0	0	0	0	1	0	1	0	0
fu	12	10	902,4	7,7	7200,0	0	1	1	1	0	1	1	1
fu5	5	4	328,7	0,0	1,9	0	0	0	0	0	1	1	0
fu6	6	4	434,0	0,0	7,4	0	0	0	0	0	0	0	0
fu7	7	5	443,9	0,0	343,0	0	0	0	0	1	0	0	1
fu8	8	6	549,1	1,4	7200,0	0	0	0	0	0	0	0	2
fu9	9	8	651,0	2,0	7200,0	0	0	0	0	0	0	0	0
fu10	10	8	796,8	5,5	7200,0	0	0	0	0	0	0	0	1
poly1a	15	11	317,9	16,7	7200,0	0	0	0	1	0	0	0	1
poly1b	15	10	319,5	18,5	7200,0	0	0	0	0	0	0	0	0
poly1c	15	15	315,5	0,0	350,5	0	0	0	0	0	0	0	0
poly1d	15	12	265,5	22,0	7200,0	0	0	0	0	0	0	0	0
poly1e	15	12	241,5	6,2	7200,0	0	0	0	0	0	0	0	0
rco1	7	6	62,1	0,0	7,5	0	0	0	1	0	1	1	2
rco2	14	13	157,6	0,0	2094,7	0	0	0	0	0	0	1	1
rco3	21	19	239,2	1,4	7200,0	0	0	0	0	0	0	0	0
rco4	28	25	337,5	2,7	7200,0	0	0	0	0	0	0	0	0
rco5	35	31	415,6	4,0	7200,0	0	0	0	0	0	0	0	0
shapes2	8	8	316,3	0,0	13,1	0	0	0	0	0	0	0	0
shapes4	16	11	377,5	1,4	7200,0	0	0	0	0	0	0	0	0
shapes5	20	12	514,5	4,3	7200,0	0	0	0	0	0	0	0	0
shapes7	28	19	789,3	6,6	7200,0	0	0	0	0	0	0	0	0
shapes9	34	23	904,0	12,8	7200,0	0	0	0	0	0	0	0	0
shapes15	43	29	1057,0	21,8	7200,0	0	0	0	0	0	0	0	0
shirts1-2	13	13	275,0	0,0	6,1	0	0	0	0	0	0	0	0
shirts2-4	26	22	440,4	0,2	7200,0	0	0	0	0	0	0	0	0
shirts3-6	39	37	716,0	1,7	7200,0	0	0	0	0	0	0	0	0
shirts4-8	52	49	863,5	25,7	7200,0	0	0	0	0	0	0	0	0
shirts5-10	65	61	1126,3	22,1	7200,0	0	0	0	0	0	0	0	0
three	3	1	3,9	0,0	0,0	0	0	0	1	0	0	1	1
threep2	6	4	20,2	0,0	0,1	0	1	1	1	0	2	1	2
threep2w9	6	4	23,9	0,0	0,1	0	0	0	1	1	0	1	2
threep3	9	5	37,3	0,0	0,4	0	0	0	0	0	1	0	2
threep3w9	9	5	38,4	0,0	0,4	0	0	0	0	0	0	1	1
Média	-	-	-	5,4	4405,2	-	-	-	-	-	-	-	-
Des. Padrão	-	13	-	7,8	3482,0	-	-	-	-	-	-	-	-

Tabela 4.4 – Resultados do caso otimista para o 2IKP-DU.

Instância	Total de itens	#Itens selecionados	Obj.	*gap* (%)	Tempo (s)	#Itens cancelados no cenário							
						#1	#2	#3	#4	#5	#6	#7	#8
blasz2	16	12	177,2	0,0	45,8	0	0	0	0	0	0	0	0
blazewicz1	7	5	53,1	0,0	5,4	0	0	0	1	0	1	1	1
blazewicz2	14	11	123,6	0,2	7200,0	0	0	0	0	0	1	0	0
blazewicz3	21	18	206,0	2,0	7200,0	0	0	0	0	0	0	0	0
blazewicz4	28	23	264,4	5,7	7200,0	0	0	0	0	0	0	0	0
blazewicz5	35	29	332,8	5,1	7200,0	0	0	0	0	0	0	0	0
dagli1	30	17	1090,6	23,5	7200,0	0	0	0	0	1	0	1	0
fu	12	10	923,9	7,3	7200,0	0	1	1	1	0	1	1	1
fu5	5	4	360,1	0,0	5,7	0	0	0	1	0	1	1	0
fu6	6	4	435,0	0,0	16,5	0	1	1	1	1	1	1	1
fu7	7	4	458,3	0,0	499,0	0	0	0	0	0	0	0	1
fu8	8	6	570,9	3,2	7200,0	0	0	0	0	0	0	0	2
fu9	9	7	680,1	0,0	4018,5	0	1	1	1	1	1	1	1
fu10	10	8	822,1	3,9	7200,0	0	0	0	0	1	1	1	1
poly1a	15	12	323,2	19,9	7200,0	1	0	0	1	1	1	1	2
poly1b	15	11	347,8	13,3	7200,0	0	1	1	0	0	1	1	1
poly1c	15	15	315,5	0,0	369,5	0	0	0	0	0	0	0	0
poly1d	15	14	280,6	16,1	7200,0	0	1	1	0	1	0	0	0
poly1e	15	13	252,3	5,1	7200,0	0	0	0	1	1	1	1	1
rco1	7	6	71,3	0,0	8,3	0	0	0	1	0	1	1	2
rco2	14	12	163,7	0,0	7200,0	0	0	0	0	0	0	0	0
rco3	21	19	245,6	1,5	7200,0	0	0	0	0	0	0	0	1
rco4	28	26	339,2	3,4	7200,0	0	0	0	0	0	0	0	0
rco5	35	31	416,1	9,0	7200,0	0	0	0	0	0	0	0	0
shapes2	8	8	318,6	0,0	69,0	0	0	0	0	0	0	0	0
shapes4	16	11	384,1	0,6	7200,0	0	0	0	0	0	0	0	0
shapes5	20	12	526,4	3,6	7200,0	0	0	0	0	0	0	0	2
shapes7	28	19	801,8	6,7	7200,0	0	0	0	0	0	0	0	0
shapes9	34	23	900,1	15,4	7200,0	0	0	0	0	0	0	0	0
shapes15	43	29	1065,8	21,8	7200,0	0	0	0	0	0	0	0	0
shirts1-2	13	13	275,0	0,0	6,0	0	0	0	0	0	0	0	0
shirts2-4	26	22	443,3	0,2	7200,0	0	0	0	0	0	0	0	0
shirts3-6	39	37	716,0	2,3	7200,0	0	0	0	0	0	0	0	0
shirts4-8	52	49	933,2	7,9	7200,0	0	0	0	0	0	0	0	0
shirts5-10	65	62	1117,0	23,1	7200,0	0	0	0	0	0	0	0	0
three	3	2	9,6	0,0	0,0	0	0	0	2	1	1	2	2
threep2	6	4	27,2	0,0	0,1	0	0	0	1	0	1	0	1
threep2w9	6	4	28,9	0,0	0,2	0	0	0	1	1	0	1	2
threep3	9	6	42,7	0,0	0,3	0	1	1	0	1	1	1	2
threep3w9	9	6	42,3	0,0	0,4	0	0	0	0	1	0	1	1
Média	-	-	-	5,0	4626,1	-	-	-	-	-	-	-	-
Des. Padrão	-	-	-	7,2	3421,3	-	-	-	-	-	-	-	-

Tabela 4.5 – Resultados do caso pessimista para o 2IKP-DU.

Instância	Total de itens	#Itens selecionados	Obj.	*gap* (%)	Tempo (s)	#Itens cancelados no cenário							
						#1	#2	#3	#4	#5	#6	#7	#8
blasz2	16	11	162,0	0,0	27,7	0	0	0	0	0	0	0	0
blazewicz1	7	5	42,6	0,0	0,7	0	0	0	1	0	0	1	1
blazewicz2	14	11	107,4	0,0	136,0	0	0	0	0	0	0	1	1
blazewicz3	21	17	196,4	0,8	7200,0	0	0	0	0	0	0	0	1
blazewicz4	28	23	262,0	1,7	7200,0	0	0	0	0	0	0	0	0
blazewicz5	35	28	328,2	4,4	7200,0	0	0	0	0	0	0	0	0
dagli1	30	14	1110,9	16,0	7200,0	0	0	0	0	0	0	0	1
fu	12	8	887,0	6,2	7200,0	0	0	0	0	0	0	0	0
fu5	5	4	296,7	0,0	1,5	0	0	0	0	0	0	1	1
fu6	6	4	434,0	0,0	6,9	0	0	0	0	0	0	0	0
fu7	7	4	441,0	0,0	212,4	0	0	0	0	0	0	0	0
fu8	8	6	515,7	0,0	612,1	0	0	0	0	0	0	0	1
fu9	9	8	651,0	0,0	796,8	0	0	0	0	0	0	0	0
fu10	10	8	775,0	5,3	7200,0	0	0	0	0	0	0	0	1
poly1a	15	10	304,0	19,4	7200,0	0	0	0	0	0	0	0	0
poly1b	15	10	321,5	18,0	7200,0	0	0	0	0	0	0	0	0
poly1c	15	15	315,5	0,0	355,1	0	0	0	0	0	0	0	0
poly1d	15	12	268,5	17,8	7200,0	0	0	0	0	0	0	0	0
poly1e	15	13	237,5	4,3	7200,0	0	0	0	0	0	0	0	0
rco1	7	5	50,9	0,0	3,8	0	0	0	1	0	0	0	1
rco2	14	12	157,0	0,0	19,2	0	0	0	0	0	0	0	0
rco3	21	18	234,0	0,1	7200,0	0	0	0	0	0	0	0	0
rco4	28	25	337,2	3,4	7200,0	0	0	0	0	0	0	0	0
rco5	35	31	414,3	6,3	7200,0	0	0	0	0	0	0	0	0
shapes2	8	8	315,8	0,0	20,8	0	0	0	0	0	0	0	0
shapes4	16	11	376,2	0,0	3362,6	0	0	0	0	0	0	0	0
shapes5	20	12	496,0	3,4	7200,0	0	0	0	0	0	0	0	0
shapes7	28	18	782,8	6,5	7200,0	0	0	0	0	0	1	0	0
shapes9	34	23	913,2	10,3	7200,0	0	0	0	0	0	0	0	0
shapes15	43	29	1062,3	20,3	7200,0	0	0	0	0	0	0	0	0
shirts1-2	13	13	275,0	0,0	6,0	0	0	0	0	0	0	0	0
shirts2-4	26	21	437,5	0,1	7200,0	0	0	0	0	0	0	0	0
shirts3-6	39	37	716,0	1,2	7200,0	0	0	0	0	0	0	0	0
shirts4-8	52	49	885,0	20,6	7200,0	0	0	0	0	0	0	0	0
shirts5-10	65	62	1138,5	20,2	7200,0	0	0	0	0	0	0	0	0
three	3	0	0,0	0,0	0,0	0	0	0	0	0	0	0	0
threep2	6	3	13,2	0,0	0,1	0	0	0	0	0	1	0	1
threep2w9	6	3	14,7	0,0	0,2	0	0	0	0	1	0	0	1
threep3	9	5	30,9	0,0	0,2	0	0	0	0	0	1	0	2
threep3w9	9	5	34,7	0,0	0,2	0	0	0	0	0	0	1	1
Média	-	-	-	4,7	3919,1	-	-	-	-	-	-	-	-
Des. Padrão	-	-	-	7,1	3533,0	-	-	-	-	-	-	-	-

Na Tabela 4.3, para o caso equiprovável, observa-se que uma solução ótima é obtida para 40,0% das instâncias, com um *gap* médio de 5,4% e tempo médio de 4405,2 segundos. A instância que apresenta o maior *gap* é a shirts4-8, com um valor de 25,7%. Nas instâncias poly1c, shapes2 e shirts1-2 é possível selecionar e produzir todos os itens, ou seja, nenhum cancelamento ocorre para qualquer cenário. Em geral, não há cancelamento de itens em qualquer cenário para 55,0% das instâncias. Por outro lado, nas instâncias com itens cancelados, o cancelamento ocorre em maior quantidade para os cenários #4, #6, #7 e #8, que são cenários com maior quantidade de defeitos. A instância com o maior cancelamento de itens é a threep2. Em comparação com o caso moderado, observa-se para o caso equiprovável menos instâncias resolvidas na otimalidade, *gap* maior, tempo computacional maior, mais cancelamento de itens por instância e cenários. Por outro lado, no caso equiprovável, as soluções em geral têm melhores valores de função objetivo e mais itens selecionados, justificando-se pela igual probabilidade de ocorrência dos cenários.

Observando a Tabela 4.4, para o caso otimista, uma solução ótima é obtida para 40,0% das instâncias, com um *gap* médio de 5,0% e tempo médio de 4626,1 segundos. A instância com maior *gap* é a dagli1, de 23,5%. Nas instâncias poly1c, shapes2 e shirts1-2 é possível selecionar todos os itens e não há qualquer cancelamento olhando os cenários. Não há qualquer cancelamento de itens para 45,0% das instâncias. Por outro lado, nas instâncias com itens cancelados, o cancelamento ocorre em maior quantidade para os cenários de #4 a #8. A instância com o maior cancelamento de itens é a three. Ao comparar os valores do caso otimista com o moderado, nota-se para o caso otimista que menos instâncias são resolvidas na otimalidade, o tempo computacional é maior, há mais cancelamento de itens por instância e cenários. Por outro lado, no caso otimista, as soluções em geral tem melhores valores de função objetivo e mais itens selecionados, justificando-se pela menor probabilidade de ocorrência dos cenários com mais defeitos.

Para o caso pessimista, na Tabela 4.5, o *gap* e o tempo médio são de 4,7% e 3919,1 segundos, respectivamente. Soluções ótimas são obtidas para 47,5% das instâncias, com o maior *gap* de 20,6% para a instância shirts4-8. Não há qualquer cancelamento de itens para 67,5% instâncias, incluindo as instâncias poly1c, shapes2 e shirts1-2, que também possuem todos os itens selecionados e produzidos. O maior número de cancelamentos ocorre no cenário #8, enquanto as instâncias blazewicz1 e threep3 têm mais itens cancelados. Em comparação com o caso moderado, observa-se para o caso pessimista mais instâncias resolvidas na otimalidade, *gap* menor, tempo computacional menor, menos cancelamento de itens por instância e cenários. Por outro lado, no caso pessimista, as soluções em geral têm piores valores de função objetivo e menos itens selecionados, justificando-se pela maior probabilidade de ocorrência dos cenários com mais defeitos.

As análises EVPI e VSS consideram 14 instâncias para as quais foi possível obter uma solução ótima em todos o casos dentro do tempo limite de 7200 segundos, como reportado nas Tabelas 4.3 a 4.5. Os resultados para o EVPI são apresentados nas Tabelas 4.6 e 4.7, para

os quatro casos: moderado, equiprovável, otimista e pessimista. Todas as soluções reportadas, incluindo aquelas para os problemas *wait-and-see*, são ótimas. O cálculo do EVPI relativo (%) é dado por $100 \times (\text{EVPI}/\text{RP})$.

Tabela 4.6 – EVPI para instâncias com solução ótima em todos os casos - parte 1.

	Moderado	Equiprovável	Otimista	Pessimista	Moderado	Equiprovável	Otimista	Pessimista
Cenário	WS_s^* para blasz2				WS_s^* para blazewicz1			
1	11,7	22,8	76,8	2,8	4,1	8,0	27,0	1,0
2	17,5	22,8	25,6	8,5	5,4	7,0	7,9	2,6
3	17,5	22,8	25,6	8,5	5,4	7,0	7,9	2,6
4	23,3	20,3	7,6	22,8	6,9	6,0	2,3	6,8
5	17,5	22,8	25,6	8,5	5,7	7,4	8,4	2,8
6	24,0	20,8	7,8	23,4	7,9	6,8	2,6	7,7
7	24,0	20,8	7,8	23,4	7,0	6,1	2,3	6,8
8	35,0	20,3	2,5	68,3	8,8	5,1	0,6	17,1
WS	170,3	173,1	179,3	166,4	51,1	53,4	58,8	47,4
RP	164,4	167,6	177,2	162,0	46,9	49,1	53,1	42,6
EVPI	6,0	5,6	2,1	4,4	4,2	4,3	5,7	4,7
EVPI (%)	3,6	3,3	1,2	2,7	8,9	8,7	10,7	11,1
	WS_s^* para fu5				WS_s^* para fu6			
1	24,9	48,6	164,1	6,1	31,0	60,6	204,6	7,6
2	37,3	48,6	54,7	18,2	41,7	54,3	61,0	20,3
3	32,5	42,3	47,5	15,8	46,6	60,6	68,2	22,7
4	54,0	46,9	17,6	52,7	62,5	54,3	20,3	61,0
5	37,3	48,6	54,7	18,2	41,7	54,3	61,0	20,3
6	48,7	42,3	15,8	47,5	62,5	54,3	20,3	61,0
7	46,9	40,8	15,3	45,8	62,5	54,3	20,3	61,0
8	62,4	36,1	4,5	121,9	93,7	54,3	6,8	183,1
WS	344,1	354,1	374,3	326,4	442,2	446,7	462,7	437,2
RP	314,4	328,7	360,1	296,7	434,0	434,0	435,0	434,0
EVPI	29,7	25,4	14,2	29,7	8,2	12,7	27,7	3,2
EVPI (%)	9,4	7,7	3,9	10,0	1,9	2,9	6,4	0,7
	WS_s^* para fu7				WS_s^* para poly1c			
1	31,0	60,6	204,6	7,6	20,2	39,4	133,1	4,9
2	46,6	60,6	68,2	22,7	30,3	39,4	44,4	14,8
3	42,3	55,1	62,0	20,7	30,3	39,4	44,4	14,8
4	69,8	60,6	22,7	68,2	45,4	39,4	14,8	44,4
5	45,2	58,9	66,2	22,1	30,3	39,4	44,4	14,8
6	69,8	60,6	22,7	68,2	45,4	39,4	14,8	44,4
7	69,8	60,6	22,7	68,2	45,4	39,4	14,8	44,4
8	95,3	55,1	6,9	186,1	68,2	39,4	4,9	133,1
WS	469,9	472,2	476,1	463,7	315,5	315,5	315,5	315,5
RP	441,0	443,9	458,3	441,0	315,5	315,5	315,5	315,5
EVPI	28,9	28,3	17,8	22,7	0,0	0,0	0,0	0,0
EVPI (%)	6,6	6,4	3,9	5,1	0,0	0,0	0,0	0,0

Os resultados das Tabelas 4.6 e 4.7 mostram valores relativos de EVPI superiores a 5% para as instâncias blazewicz1, three, threep2, threep2w9, threep3 e threep3w9 (em todos os casos), fu5 e fu7 (exceto no caso otimista), fu6 (somente no caso otimista) e rco1 (exceto no caso pessimista). As instâncias menos impactadas em termos de EVPI são aquelas cujos defeitos influenciam pouco nos cenários, como é o caso da poly1c, shapes2 e shirts1-2, não resultando em diferença entre resolver os problemas *wait-and-see* ou o modelo de programação estocástica de dois estágios. Por outro lado, os maiores valores relativos de EVPI ocorrem para as instâncias three, threep2 e threep2w9, com um valor médio geral de 91,7%, 22,7% e 18,8%,

Tabela 4.7 – EVPI para instâncias com solução ótima em todos os casos - parte 2.

	Moderado	Equiprovável	Otimista	Pessimista	Moderado	Equiprovável	Otimista	Pessimista
Cenário	WS_s^* para rco1				WS_s^* para shapes2			
1	5,0	9,8	33,1	1,2	20,5	40,0	135,0	5,0
2	7,2	9,3	10,5	3,5	30,7	40,0	45,0	15,0
3	7,2	9,3	10,5	3,5	30,7	40,0	45,0	15,0
4	8,4	7,3	2,7	8,2	46,1	40,0	15,0	45,0
5	7,5	9,8	11,0	3,7	30,7	40,0	45,0	15,0
6	9,3	8,1	3,0	9,1	43,2	37,5	14,1	42,2
7	10,4	9,1	3,4	10,2	46,1	40,0	15,0	45,0
8	10,3	5,9	0,7	20,0	69,1	40,0	5,0	135,0
WS	65,3	68,6	75,0	59,4	317,1	317,5	319,1	317,2
RP	57,0	62,1	71,3	59,0	315,7	316,3	318,6	315,8
EVPI	8,3	6,5	3,7	0,4	1,4	1,2	0,5	1,4
EVPI (%)	14,5	10,5	5,2	0,7	0,4	0,4	0,1	0,4
	WS_s^* para shirts1-2				WS_s^* para three			
1	17,6	34,4	116,0	4,3	1,1	2,1	7,2	0,3
2	26,4	34,4	38,7	12,9	1,4	1,9	2,1	0,7
3	26,4	34,4	38,7	12,9	0,9	1,1	1,3	0,4
4	39,6	34,4	12,9	38,7	0,0	0,0	0,0	0,0
5	26,4	34,4	38,7	12,9	0,9	1,1	1,3	0,4
6	39,6	34,4	12,9	38,7	1,3	1,1	0,4	1,3
7	39,6	34,4	12,9	38,7	0,0	0,0	0,0	0,0
8	59,4	34,4	4,3	116,0	0,0	0,0	0,0	0,0
WS	275,0	275,0	275,0	275,0	5,6	7,4	12,2	3,1
RP	275,0	275,0	275,0	275,0	2,2	3,9	9,6	0,0
EVPI	0,0	0,0	0,0	0,0	3,4	3,4	2,7	3,1
EVPI (%)	0,0	0,0	0,0	0,0	152,3	86,8	27,6	100,0
	WS_s^* para threep2				WS_s^* para threep2w9			
1	2,2	4,3	14,3	0,5	2,2	4,3	14,3	0,5
2	2,5	3,3	3,7	1,2	3,1	4,0	4,5	1,5
3	2,5	3,3	3,7	1,2	3,1	4,0	4,5	1,5
4	3,5	3,0	1,1	3,4	3,7	3,3	1,2	3,7
5	3,3	4,3	4,8	1,6	2,3	3,0	3,4	1,1
6	2,6	2,3	0,8	2,5	4,6	4,0	1,5	4,5
7	3,7	3,3	1,2	3,7	3,7	3,3	1,2	3,7
8	1,9	1,1	0,1	3,8	1,9	1,1	0,1	3,8
WS	22,2	24,6	29,8	17,9	24,7	26,9	30,8	20,3
RP	17,8	20,2	27,2	13,2	20,8	23,9	28,9	14,7
EVPI	4,3	4,4	2,5	4,7	3,9	2,9	1,9	5,6
EVPI (%)	24,3	21,9	9,3	35,3	18,5	12,2	6,4	38,1
	WS_s^* para threep3				WS_s^* para threep3w9			
1	3,1	6,1	20,7	0,8	3,1	6,1	20,7	0,8
2	4,1	5,4	6,1	2,0	3,9	5,1	5,8	1,9
3	4,3	5,6	6,3	2,1	4,1	5,4	6,1	2,0
4	5,9	5,1	1,9	5,8	5,9	5,1	1,9	5,8
5	4,1	5,4	6,1	2,0	4,3	5,6	6,3	2,1
6	5,0	4,4	1,6	4,9	5,9	5,1	1,9	5,8
7	6,5	5,6	2,1	6,3	5,0	4,4	1,6	4,9
8	5,8	3,4	0,4	11,4	7,3	4,3	0,5	14,3
WS	39,0	41,0	45,2	35,3	39,7	41,1	44,8	37,6
RP	35,2	37,3	42,7	30,9	37,1	38,4	42,3	34,7
EVPI	3,8	3,8	2,5	4,5	2,6	2,7	2,5	3,0
EVPI (%)	10,8	10,1	5,8	14,4	7,0	7,1	5,9	8,5

respectivamente. Considerando todas as instâncias, o valor médio relativo do EVPI é 18,4% (com desvio padrão de 39,2%) para o caso moderado, 12,7% (com desvio padrão de 22,1%) para o caso equiprovável, 6,2% (com desvio padrão de 7,0%) para o caso otimista e 16,2% (com desvio padrão de 27,1%) para o caso pessimista. Esses valores são relativamente altos para o EVPI, principalmente para os casos moderado, equiprovável e pessimista.

De maneira geral, os valores de EVPI nas Tabelas 4.6 e 4.7 indicam que o conhecimento futuro dos defeitos na placa seria vantajoso para tomar decisões precisas sobre quais itens selecionar e produzir. Dessa forma, a resolução do modelo de programação estocástica de dois estágios é importante para o problema, uma vez que os valores de EVPI são relativamente altos para a grande maioria das instâncias e também na média geral para os casos. Os menores valores de EVPI observados para o caso otimista se justificam pela baixa probabilidade de ocorrência dos cenários com mais defeitos.

Os resultados para o VSS são apresentados na Tabela 4.8 considerando os quatro casos em estudo (moderado, equiprovável, otimista e pessimista). Todas as soluções reportadas na tabela são ótimas. No problema de valor esperado, o cenário de referência consiste no cenário #8 da Tabela 4.2, que é aquele com maior quantidade de defeitos e poderia representar a pior situação para a tomada de decisões. Além disso, apresenta-se o desvio relativo (isto é, a porcentagem de melhora) da solução RP sobre a solução do EVV, dado por $100 \times (\text{RP} - \text{EVV})/\text{EVV}$. Em geral, os valores de VSS mostram que resolver o modelo de programação estocástica de dois estágios para o problema é mais vantajoso que resolver um problema de valor esperado, em especial para os casos otimista, equiprovável e moderado. Ou seja, o custo de ignorar as incertezas do problema é relativamente alto e isso reflete no baixo valor objetivo do problema de valor esperado.

Nos resultados da Tabela 4.8, as instâncias com maiores valores de VSS que, por consequência, apresentam maiores diferenças relativas (isto é, porcentagem de melhora) das soluções RP sobre as EVV são blazewicz1, fu5, rco1, threep2, threep2w9, threep3 e threep3w9, para todos os casos. Por exemplo, em todos os casos das instâncias rco1 e threep2, as diferenças relativas são superiores a 30%, enquanto na instância threep2w9 os valores ultrapassam os 60%. Por outro lado, não há diferença para algumas instâncias, como a poly1c, shapes2 e a shirts1-2, por se tratarem de instâncias cujo efeito dos defeitos traz pouca influência no cancelamento dos itens (ou seja, há poucos defeitos ou eles estão em posições que permitem os itens selecionados serem reposicionados dentro da placa para evitar o cancelamento).

Analisando os valores de VSS entre os casos, na Tabela 4.8, nota-se que o caso otimista apresenta a maior diferença relativa média, de 45,0%, seguido do caso equiprovável, com valor médio de 30,1%, depois pelo caso moderado, com valor médio de 23,2%, e, por fim, o caso pessimista, com valor médio de 12,1%. Os menores valores para o caso pessimista são justificados pelo fato do EVV considerar a solução advinda da resolução do problema de valor esperado com o pior cenário (isto é, o cenário #8, que tem mais defeitos). Dessa forma, a tendência é selecionar

Tabela 4.8 – VSS para as 14 instâncias com solução ótima em todos os casos.

	Obj. EVV	VSS	Dev. (%)	Obj. EVV	VSS	Dev. (%)
Instância	Moderado			Equiprovável		
blasz2	162,0	2,4	1,5	162,0	5,6	3,4
blazewicz1	40,5	6,4	15,7	40,5	8,6	21,3
fu5	289,0	25,4	8,8	289,0	39,7	13,7
fu6	434,0	0,0	0,0	434,0	0,0	0,0
fu7	441,0	0,0	0,0	441,0	2,9	0,7
poly1c	315,5	0,0	0,0	315,5	0,0	0,0
rco1	43,7	13,3	30,4	44,2	17,9	40,4
shapes2	315,7	0,0	0,0	316,3	0,0	0,0
shirts1-2	275,0	0,0	0,0	275,0	0,0	0,0
three	0,0	2,2	0,0	0,0	3,9	0,0
threep2	9,0	8,8	98,2	9,0	11,2	124,3
threep2w9	9,0	11,8	131,1	9,0	14,9	166,0
threep3	27,0	8,2	30,3	27,0	10,3	38,0
threep3w9	34,0	3,1	9,1	34,0	4,4	12,9
Média	-	-	23,2	-	-	30,1
Des. Padrão	-	-	40,7	-	-	51,3
	Otimista			Pessimista		
blasz2	162,0	15,2	9,4	162,0	0,0	0,0
blazewicz1	40,5	12,6	31,1	40,5	2,1	5,2
fu5	289,0	71,1	24,6	289,0	7,7	2,7
fu6	434,0	1,0	0,2	434,0	0,0	0,0
fu7	441,0	17,3	3,9	441,0	0,0	0,0
poly1c	315,5	0,0	0,0	315,5	0,0	0,0
rco1	46,3	25,0	54,1	43,8	15,2	34,7
shapes2	318,6	0,0	0,0	315,8	0,0	0,0
shirts1-2	275,0	0,0	0,0	275,0	0,0	0,0
three	0,0	9,6	0,0	0,0	0,0	0,0
threep2	9,0	18,2	202,6	9,0	4,2	47,1
threep2w9	9,0	19,9	221,4	9,0	5,7	63,1
threep3	27,0	15,7	58,2	27,0	3,9	14,3
threep3w9	34,0	8,3	24,5	34,0	0,7	2,0
Média	-	-	45,0	-	-	12,1
Des. Padrão	-	-	73,5	-	-	20,8

menos itens e, por conseguinte, cancelar menos itens nos cenários. Isso resulta em uma diferença menor entre a solução RP e a solução EVV, no caso pessimista, ao comparar com os demais casos.

4.4.3 Resultados do modelo averso ao risco

A resolução do modelo equivalente determinístico averso ao risco (4.11)-(4.14) inicia com o parâmetro $\alpha = 1$, para $\Delta^{\max}$ obtido da solução do modelo equivalente determinístico de risco neutro. Os resultados apresentados nas Tabelas 4.9 e 4.10 consideram as mesmas 14 instâncias usadas nos experimentos para as análises EVPI e VSS, sobre os quatro casos em estudo. A tabela contém o valor objetivo da solução ótima do modelo averso ao risco para $\alpha \in \{1; 0,75; 0,5; 0,25; 0,0\}$. Além disso, apresenta-se o desvio relativo (isto é, a porcentagem de piora) do valor objetivo de um dado α em comparação com o valor obtido para $\alpha = 1$, que corresponde a solução RP do modelo de risco neutro.

Os resultados nas Tabelas 4.9 e 4.10 mostram que a solução piora (isto é, o lucro com o corte da placa diminui) a medida que se reduz α (isto é, com o aumento da aversão ao risco). Observando todos os casos, as instâncias mais afetadas com a redução de α são blazewicz1, fu5, rco1, three, threep2, threep2w9 e threep3. Por exemplo, para as instâncias threep2 e threep3, a redução no valor objetivo é superior a 7% (para $\alpha = 0,75$) e 23% (para $\alpha = 0,25$) no caso moderado. Por outro lado, instâncias como fu6, poly1c e shirts1-2 possuem pouca ou nenhuma piora no valor objetivo com o aumento da aversão ao risco, justificando-se pela pouca ou nenhuma influência dos defeitos na seleção e posicionamento dos itens na placa.

Observando as Tabelas 4.9 e 4.10, no geral, o caso com maior redução do valor objetivo é o otimista, com um desvio relativo médio de 27,7% quando $\alpha = 0$ (isto é, assumindo um problema totalmente averso ao risco), seguido do caso equiprovável cujo desvio relativo médio é de 22,8% e do caso moderado, com um desvio relativo médio de 20,3%. O caso pessimista apresenta as menores reduções no valor objetivo a medida que se aumenta a aversão ao risco (isto é, que se reduz o valor de α), com um desvio relativo médio de 7,5% quando $\alpha = 0$. Nota-se que no caso pessimista há uma maior probabilidade de ocorrência dos cenários com mais defeitos, resultando em soluções com a seleção de menos itens (para evitar cancelamentos). Por outro lado, ao considerar uma redução de 25% ou 50% em $\Delta_{\max}$, ou seja, $\alpha = 0,75$ ou $\alpha = 0,5$, as maiores reduções ocorrem para os casos equiprovável e moderado, seguido do caso otimista.

A Figura 4.3 ilustra a redução percentual no valor objetivo das instâncias blazewicz1, fu5, rco1, threep2, threep2w9 e threep3 para todos os casos (moderado, equiprovável, otimista e pessimista). A construção da curva de cada caso considera a resolução do modelo averso ao risco para α variando em intervalos de 5% até atingir $\alpha = 0$. Observa-se que as curvas decrescem a medida que os valores de α diminuem, exaltando a impacto que o caso otimista sofre quando se considera um problema totalmente averso ao risco.

Tabela 4.9 – Soluções do modelo averso ao risco para os casos moderado e equiprovável.

Instância	$1,0 \times \Delta_{max}$	$0,75 \times \Delta_{max}$		$0,5 \times \Delta_{max}$		$0,25 \times \Delta_{max}$		$0,0 \times \Delta_{max}$	
	Obj.	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)
	Moderado								
blasz2	164,4	162,0	-1,4	162,0	-1,4	162,0	-1,4	162,0	-1,4
blazewicz1	46,9	45,1	-3,9	40,8	-13,1	40,5	-13,6	40,5	-13,6
fu5	314,4	311,5	-0,9	291,6	-7,3	289,0	-8,1	289,0	-8,1
fu6	434,0	434,0	0,0	434,0	0,0	434,0	0,0	434,0	0,0
fu7	441,0	441,0	0,0	441,0	0,0	441,0	0,0	441,0	0,0
poly1c	315,5	315,5	0,0	315,5	0,0	315,5	0,0	315,5	0,0
rco1	57,0	55,2	-3,2	54,8	-3,9	47,0	-17,5	47,0	-17,5
shapes2	315,7	300,0	-5,0	300,0	-5,0	300,0	-5,0	300,0	-5,0
shirts1-2	275,0	275,0	0,0	275,0	0,0	275,0	0,0	275,0	0,0
three	2,2	1,5	-33,6	0,0	-100,0	0,0	-100,0	0,0	-100,0
threep2	17,8	15,1	-15,5	13,1	-26,8	9,0	-49,6	9,0	-49,6
threep2w9	20,8	19,1	-8,0	15,1	-27,5	9,0	-56,7	9,0	-56,7
threep3	35,2	32,4	-7,8	31,1	-11,7	27,0	-23,2	27,0	-23,2
threep3w9	37,1	34,0	-8,4	34,0	-8,4	34,0	-8,4	34,0	-8,4
Média	-	-	-6,3	-	-14,6	-	-20,3	-	-20,3
Des. Padrão	-	-	9,1	-	26,2	-	29,3	-	29,3
	Equiprovável								
blasz2	167,6	163,4	-2,5	162,0	-3,3	162,0	-3,3	162,0	-3,3
blazewicz1	49,1	46,9	-4,6	45,0	-8,4	40,5	-17,5	40,5	-17,5
fu5	328,7	319,6	-2,8	301,1	-8,4	289,0	-12,1	289,0	-12,1
fu6	434,0	434,0	0,0	434,0	0,0	434,0	0,0	434,0	0,0
fu7	443,9	441,0	-0,6	441,0	-0,6	441,0	-0,6	441,0	-0,6
poly1c	315,5	315,5	0,0	315,5	0,0	315,5	0,0	315,5	0,0
rco1	62,1	58,4	-5,9	57,3	-7,8	52,4	-15,7	47,0	-24,3
shapes2	316,3	300,0	-5,1	300,0	-5,1	300,0	-5,1	300,0	-5,1
shirts1-2	275,0	275,0	0,0	275,0	0,0	275,0	0,0	275,0	0,0
three	3,9	2,6	-33,5	0,0	-100,0	0,0	-100,0	0,0	-100,0
threep2	20,2	20,1	-0,6	16,3	-19,2	13,9	-31,3	9,0	-55,4
threep2w9	23,9	21,2	-11,5	16,3	-31,9	13,9	-42,0	9,0	-62,4
threep3	37,3	33,5	-10,1	33,5	-10,1	27,0	-27,5	27,0	-27,5
threep3w9	38,4	34,0	-11,4	34,0	-11,4	34,0	-11,4	34,0	-11,4
Média	-	-	-6,3	-	-14,7	-	-19,0	-	-22,8
Des. Padrão	-	-	8,9	-	26,0	-	26,7	-	29,9

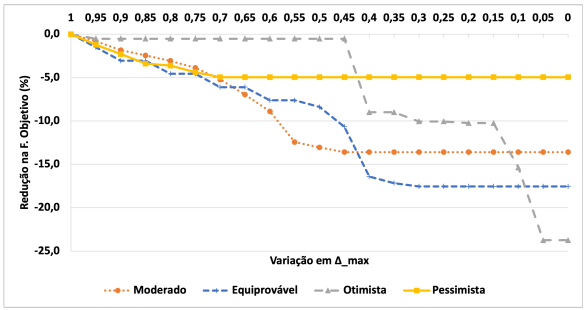
4.5 Considerações finais

Apresenta-se um modelo de programação estocástica de dois estágios e risco neutro para o problema da mochila com itens irregulares e incertezas associadas aos defeitos da placa. O primeiro estágio do modelo decide quais itens selecionar, enquanto o segundo estágio lida com as decisões para definir um plano de corte viável conforme a realização dos defeitos na placa. Enquanto esse modelo é de risco neutro, fez-se a sua extensão para considerar uma medida de aversão ao risco que limita a variabilidade da solução de segundo estágio, visando obter soluções robustas. Os modelos são resolvidos pelo algoritmo *branch-and-cut* do Gurobi, fazendo uso das ferramentas *no-fit raster* e *inner-fit raster* para garantir um posicionamento viável dos itens na placa.

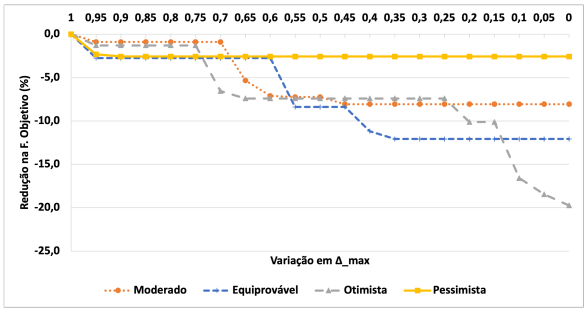
Tabela 4.10 – Soluções do modelo averso ao risco para os casos otimista e pessimista.

Instância	$1,0 \times \Delta_{max}$	$0,75 \times \Delta_{max}$		$0,5 \times \Delta_{max}$		$0,25 \times \Delta_{max}$		$0,0 \times \Delta_{max}$	
	Obj.	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)	Obj.	Dev. (%)
	Otimista								
blasz2	177,2	174,5	-1,5	165,5	-6,6	162,0	-8,6	162,0	-8,6
blazewicz1	53,1	52,8	-0,5	52,8	-0,5	47,8	-10,1	40,5	-23,7
fu5	360,1	355,4	-1,3	333,4	-7,4	333,4	-7,4	289,0	-19,7
fu6	435,0	434,0	-0,2	434,0	-0,2	434,0	-0,2	434,0	-0,2
fu7	458,3	441,0	-3,8	441,0	-3,8	441,0	-3,8	441,0	-3,8
poly1c	315,5	315,5	0,0	315,5	0,0	315,5	0,0	315,5	0,0
rco1	71,3	66,2	-7,1	60,3	-15,5	58,2	-18,4	47,0	-34,1
shapes2	318,6	300,0	-5,8	300,0	-5,8	300,0	-5,8	300,0	-5,8
shirts1-2	275,0	275,0	0,0	275,0	0,0	275,0	0,0	275,0	0,0
three	9,6	7,5	-21,6	7,5	-21,6	0,0	-100,0	0,0	-100,0
threep2	27,2	27,0	-1,0	24,5	-10,1	23,2	-14,7	9,0	-66,9
threep2w9	28,9	23,9	-17,5	23,7	-18,3	23,7	-18,3	9,0	-68,9
threep3	42,7	41,5	-2,9	40,3	-5,8	37,3	-12,7	27,0	-36,8
threep3w9	42,3	40,4	-4,6	40,4	-4,6	40,4	-4,6	34,0	-19,7
Média	-	-	-4,8	-	-7,2	-	-14,6	-	-27,7
Des. Padrão	-	-	6,7	-	6,9	-	25,4	-	30,9
	Pessimista								
blasz2	162,0	162,0	0,0	162,0	0,0	162,0	0,0	162,0	0,0
blazewicz1	42,6	40,7	-4,4	40,5	-5,0	40,5	-5,0	40,5	-5,0
fu5	296,7	289,0	-2,6	289,0	-2,6	289,0	-2,6	289,0	-2,6
fu6	434,0	434,0	0,0	434,0	0,0	434,0	0,0	434,0	0,0
fu7	441,0	441,0	0,0	441,0	0,0	441,0	0,0	441,0	0,0
poly1c	315,5	315,5	0,0	315,5	0,0	315,5	0,0	315,5	0,0
rco1	50,9	48,1	-5,4	47,0	-7,6	47,0	-7,6	47,0	-7,6
shapes2	315,8	300,0	-5,0	300,0	-5,0	300,0	-5,0	300,0	-5,0
shirts1-2	275,0	275,0	0,0	275,0	0,0	275,0	0,0	275,0	0,0
three	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
threep2	13,2	12,3	-7,1	11,2	-15,4	9,0	-32,0	9,0	-32,0
threep2w9	14,7	12,5	-15,2	11,9	-18,7	9,0	-38,7	9,0	-38,7
threep3	30,9	29,9	-3,0	29,2	-5,4	27,0	-12,5	27,0	-12,5
threep3w9	34,7	34,0	-1,9	34,0	-1,9	34,0	-1,9	34,0	-1,9
Média	-	-	-3,2	-	-4,4	-	-7,5	-	-7,5
Des. Padrão	-	-	4,2	-	6,0	-	12,4	-	12,4

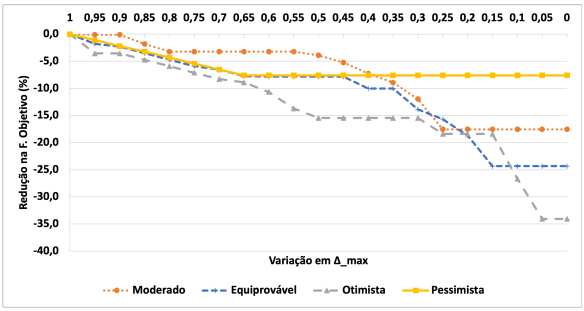
Os testes computacionais com o modelo de risco neutro mostram que os casos onde a probabilidade de ocorrência dos cenários com menos defeitos é maior tendem a ter melhores valores objetivo (isto é, resultam em maior lucro ao cortar a placa), como no caso otimista. As soluções do caso pessimista consideram a seleção de menos itens para evitar os custos de cancelamento já que os cenários com mais defeitos têm maiores chances de ocorreram. Por sua vez, as análises do valor esperado da informação perfeita e do valor da solução estocástica indicam que as incertezas sobre os defeitos impactam diretamente no lucro ao cortar a placa, sendo realmente vantajoso resolver um modelo de programação estocástica do que considerar uma solução aproximada, por resolver problemas *wait-and-see* ou um problema de valor esperado. Por outro lado, ao avaliar as soluções do modelo averso ao risco, observa-se uma piora significativa do valor objetivo quando se tem um decisor totalmente conservador, em particular, para o caso



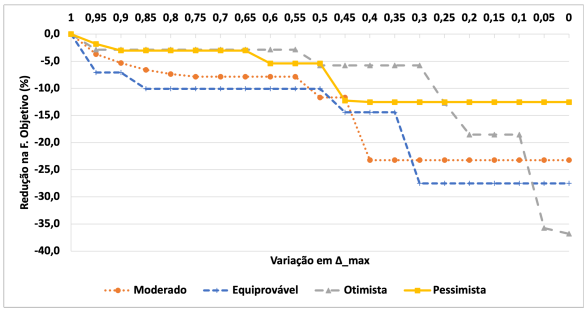
(a) blazewicz1



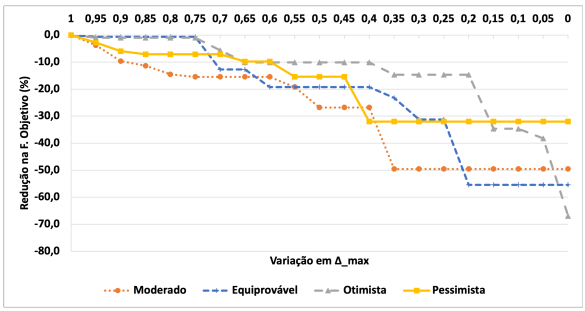
(b) fu5



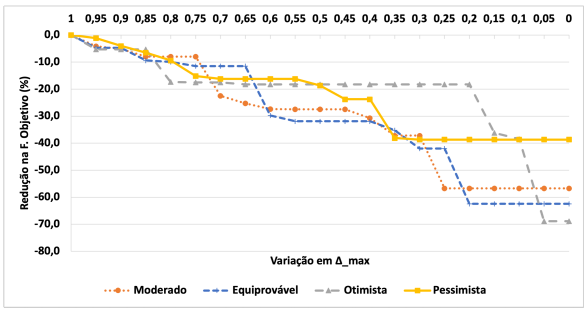
(c) rco1



(d) threep3



(e) threep2



(f) threep2w9

Figura 4.3 – Redução percentual do valor objetivo de instâncias das Tabelas 4.9 e 4.10 considerando α reduzindo de 5% em 5%.

otimista.

CAPÍTULO

5

CONCLUSÕES E PESQUISAS FUTURAS

O estudo de problemas de corte e empacotamento de itens irregulares, ou seja, problemas de *nesting*, é justificado pelo significativo número de aplicações e pela dificuldade adicional que envolve a geometria dos itens. A literatura sobre esses problemas tem proposto métodos de solução que vão desde a resolução de modelos de programação linear inteira e de programação por restrições, as mais variadas heurísticas, como construtivas, de busca local e meta-heurísticas. Apesar dos esforços já desprendidos pela literatura, nota-se que ainda há espaço para propor novos métodos de solução, bem como incluir e modelar características e restrições que advêm de contextos reais, como a presença de incertezas. Essas duas linhas foram investigadas nesta tese, seja pela proposta de novos métodos de solução, mas também pelo estudo de características oriundas de contextos reais, em particular, a consideração de incertezas nos problemas.

A contribuição na linha de novos métodos de solução foi dada no Capítulo 2, com a proposta de duas heurísticas para o problema da mochila (sem incertezas), um algoritmo genético de chaves aleatórias viciadas e uma busca em vizinhança variável. Nessas heurísticas, o vetor que codifica a solução tem informações sobre a sequência e a rotação em que os itens deveriam ser posicionados no recipiente. O vetor também contém informação sobre a quantidade de posições viáveis que devem ser ignoradas durante o posicionamento de um item. No algoritmo genético, o vetor ainda contém a probabilidade, caso o cromossomo seja elite, de suas informações serem passadas aos descendentes. O posicionamento dos itens no recipiente é feito por três regras, sendo uma delas a *bottom-left* e duas outras derivadas dela. Os resultados computacionais indicaram a superioridade das heurísticas propostas, inclusive havendo significância estatística, já que obtiveram soluções iguais ou melhores em todas as instâncias. Um marco importante foi a capacidade das heurísticas em melhorar a ocupação do recipiente em comparação com o estado-da-arte do problema. Enquanto a busca em vizinhança variável permitiu um aumento na ocupação do recipiente de 5,9% na média, o algoritmo genético conseguiu ter uma média de 6,4%. Observa-se que o algoritmo genético foi capaz de obter melhores soluções e em menor tempo computacional por trabalhar sobre uma população de indivíduos, permitindo gerar diversas

sequências de itens, e pelo fato de se ter implementado a probabilidade de cruzamento como parte do indivíduo ao invés de ser um parâmetro constante. Trabalhos futuros que podem ser realizados a partir das contribuições desse capítulo dizem respeito a:

- estender as heurísticas para abordar os problemas de uma ou de duas dimensões abertas. A literatura do problema de duas dimensões abertas ainda é mais limitada que a literatura do problema da mochila, sendo uma oportunidade para apresentar métodos que poderão se tornar estado-da-arte para esse problema;
- propor outras buscas locais para a busca em vizinhança variável, por exemplo, usando apenas uma das estruturas de vizinhança. Notou-se que essa heurística gastava bastante tempo na etapa da busca local, grande parte das vezes sem conseguir melhorar a solução;
- desenvolver e testar novas regras de posicionamento, sejam aquelas baseadas na *bottom-left*, mas como outras que permitam melhor combinar os itens e evitar espaços vazios no recipiente;
- combinar a resolução de modelos de programação matemática com as heurísticas propostas, visando obter mais soluções ótimas para o problema.

O Capítulo 4 contribui nas duas linhas, de novos métodos de solução e do estudo de características oriundas de contextos reais, em particular, o estudo de incertezas. Nesse capítulo se estuda o problema de corte em faixa com os itens apresentando incertezas nas suas demandas. Apresenta-se para o problema um modelo de programação estocástica de dois estágios com recurso. No primeiro estágio, busca-se definir a área inicial de faixa a comprar. No segundo estágio, obtém-se o custo esperado pela possível compra de área adicional de faixa em conformidade com a realização das demandas dos itens a partir de cenários. A resolução desse modelo é feita por um algoritmo *branch-and-cut*, que incorpora uma heurística baseada em busca em vizinhança variável. Essa heurística gera sequências de itens posicionados por duas heurísticas, sendo a *bottom-left* uma delas. A partir das soluções fracionárias nos nós da árvore de busca, a heurística gera uma solução viável que é, então, usada para definir limitantes válidos. O algoritmo proposto apresentou resultados competitivos com outros dois métodos exatos da literatura para o problema sem incertezas, resolvendo na otimalidade cerca de 57% das instâncias e apresentando um *gap* médio de 8,1%. Já na presença do problema com incertezas, o algoritmo foi capaz de obter a solução ótima para cerca de 35% das instâncias contendo 9 cenários. Ele também foi capaz de resolver otimamente instâncias com mais de 60 cenários. As análises do valor esperado da informação perfeita indicaram um alto desvio relativo entre as soluções, com o valor médio variando entre cerca de 70% e 86%. Igualmente, as análises do problema de valor esperado retornaram um desvio relativo médio variando de 5% a 35%. Essas análises indicam pela necessidade de se considerar as incertezas no problema para que sejam obtidas soluções

de baixo custo. Trabalhos futuros que podem ser desenvolvidos a partir dos resultados desse capítulo são:

- adaptar a heurística de busca em vizinhança variável para resolver o problema estocástico e comparar suas soluções com o algoritmo *branch-and-cut*;
- fazer uso de métodos de decomposição, buscando aproveitar a estrutura favorável que os modelos de programação estocástica possuem;
- considerar outras características presentes nas situações reais, como permitir a rotação dos itens e a faixa com zonas de qualidade;
- desenvolver modelos de otimização robusta, bem como avaliar a aplicação de outras metodologias para uma adequada representação das incertezas no problema.

Por fim, outra contribuição na linha de características oriundas de contextos reais é dada no Capítulo 4, com um modelo de programação estocástica de dois estágios com recurso para o problema da mochila com incertezas associadas aos defeitos do recipiente. O primeiro estágio do modelo está relacionado com a seleção de itens para a eventual produção. No segundo estágio do modelo, decide-se o plano de corte do recipiente conforme a realização dos defeitos a partir de cenários amostrados. Sendo esse um modelo de risco neutro, ou seja, cuja solução é para longo prazo, fez-se a sua extensão para obter soluções robustas, ou seja, com a inclusão de uma medida de risco. Essa medida limita a variabilidade da solução de segundo estágio. Os testes computacionais com o modelo de risco neutro indicaram que os casos cujos cenários com menos defeitos têm maior probabilidade de ocorrência possuem soluções de maior lucro. Nas análises do valor esperado da informação perfeita e do valor da solução estocástica, percebe-se claramente que as incertezas sobre os defeitos impactam no lucro das soluções. Isso significa que é melhor resolver um modelo de programação estocástica do que aproximar a solução do problema por cenários específicos ou um problema de valor esperado. Por outro lado, se o desejo é obter soluções robustas (isto é, aversas ao risco), nota-se pelas soluções do modelo averso ao risco que o lucro ao cortar o recipiente pode ser reduzido de forma significativa. A partir desse estudo, apontam-se as seguintes direções para trabalhos futuros:

- avaliar os modelos na presença de mais defeitos na placa e para um número maior de cenários, já que algumas instâncias foram pouco impactadas pelos defeitos e cenários propostos;
- propor heurísticas que possam resolver o problema estocástico e, assim, comparar com as soluções dos modelos de programação estocástica. Uma alternativa inicial seria adaptar as heurísticas desenvolvidas no Capítulo 2;

- acelerar a resolução dos modelos de programação estocástica com o uso de métodos de decomposição. Modelos de programação estocástica possuem uma estrutura favorável para a aplicação desses métodos, pois definidas as variáveis de primeiro estágio, o modelo equivalente determinístico pode ser decomposto em modelos independentes conforme o número de cenários (BIRGE; LOUVEAUX, 1997).

REFERÊNCIAS

ABEYSOORIYA, R. P. **Cutting Patterns for Efficient Production of Irregular-shaped Pieces**. 203 p. Tese (Doctor of Philosophy in Operational Research) — Faculty of Business, Law and Art, University of Southampton, Southampton, United Kingdom, 2017. Citado na página 33.

ABEYSOORIYA, R. P.; BENNELL, J. A.; SYKORA, A. M. Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation. **International Journal of Production Economics**, v. 195, p. 12–26, 2018. Citado na página 37.

AHMED, S.; SAHINIDIS, N. V. Robust process planning under uncertainty. **Industrial & Engineering Chemistry Research**, v. 37, n. 5, p. 1883–1892, 1998. Citado nas páginas 105 e 110.

AIEX, R. M.; BINATO, S.; RESENDE, M. G. C. Parallel grasp with path-relinking for job shop scheduling. **Parallel Computing**, v. 29, n. 4, p. 393–430, 2003. Citado na página 63.

ALBANO, A.; SAPUPPO, G. Optimal allocation of two-dimensional irregular shapes using heuristic search methods. **IEEE Transactions on Systems, Man, and Cybernetics**, v. 10, n. 5, p. 242–248, 1980. Citado nas páginas 30, 35, 46 e 75.

ALEM, D.; MORABITO, R. Modelos de programação estocástica no planejamento da produção de empresas moveleiras. **Production [online]**, v. 25, n. 3, p. 657–677, 2015. Citado nas páginas 89, 110 e 112.

____. Planejamento da produção sob incerteza: programação estocástica versus otimização robusta. **Gestão & Produção**, v. 22, n. 3, p. 539–551, 2015. Citado na página 38.

ALEM, D. J.; MUNARI, P. A.; ARENALES, M. N.; FERREIRA, P. A. V. On the cutting stock problem under stochastic demand. **Annals of Operations Research**, v. 179, n. 1, p. 169–186, 2010. Citado na página 27.

ALVAREZ-VALDES, R.; MARTINEZ, A.; TAMARIT, J. M. A branch & bound algorithm for cutting and packing irregularly shaped pieces. **International Journal of Production Economics**, v. 145, n. 2, p. 463–477, 2013. Citado nas páginas 34, 62, 65, 75, 88 e 112.

ALVES, C. M. M.; BRÁS, P. A. F.; CARVALHO, J. M. V.; PINTO, T. M. P. New constructive algorithms for leather nesting in the automotive industry. **Computers & Operations Research**, v. 39, n. 7, p. 1487–1505, 2012. Citado nas páginas 38, 48 e 104.

____. A variable neighborhood search algorithm for the leather nesting problem. **Mathematical Problems in Engineering**, v. 2012, p. 254346, 2012. Citado nas páginas 38, 46, 47, 48 e 104.

AMARO JR., B.; PINHEIRO, P. R.; COELHO, P. V. A parallel biased random-key genetic algorithm with multiple populations applied to irregular strip packing problems. **Mathematical Problems in Engineering**, v. 2017, p. 1670709, 2017. Citado nas páginas 36 e 48.

ANDRADE, C. E.; RESENDE, M. G. C.; ZHANG, W.; SINHA, R. K.; REICHMANN, K. C.; DOVERSPIKE, R. D.; MIYAZAWA, F. K. A biased random-key genetic algorithm for wireless backhaul network design. **Applied Soft Computing**, v. 33, p. 150–169, 2015. Citado na página 48.

AURELIANO, F. A. **Estudo de métodos de solução para problemas de corte de itens irregulares em recipientes irregulares**. 91 p. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2017. Citado na página 33.

BAKER, B.; COFFMAN JR., E.; RIVEST, R. Orthogonal packings in two dimensions. **SIAM Journal on Computing**, v. 9, n. 4, p. 846–855, 1980. Citado na página 35.

BALDACCI, R.; BOSCHETTI, M. A.; GANOVELLI, M.; MANIEZZO, V. Algorithms for nesting with defects. **Discrete Applied Mathematics**, v. 163, Part 1, p. 17–33, 2014. Citado nas páginas 33, 46, 47, 102 e 103.

BELLMAN, R. **Dynamic programming**. [S.l.]: Princeton University Press, 1957. Citado na página 39.

BELLMAN, R. E.; ZADEH, L. A. Decision making in fuzzy environment. **Management Science**, v. 17, p. 141–154, 1970. Citado na página 39.

BEN-TAL, A.; NEMIROVSKI, A. Robust convex optimization. **Mathematics of Operations Research**, v. 23, n. 4, p. 769–805, 1998. Citado na página 26.

BENNELL, J.; SCHEITHAUER, G.; STOYAN, Y.; ROMANOVA, T.; PANKRATOV, A. Optimal clustering of a pair of irregular objects. **Journal of Global Optimization**, v. 61, n. 3, p. 497–524, 2015. Citado nas páginas 36 e 46.

BENNELL, J. A.; CABO, M.; SYKORA, A. M. A beam search approach to solve the convex irregular bin packing problem with guillotine cuts. **European Journal of Operational Research**, v. 270, n. 1, p. 89–102, 2018. Citado na página 37.

BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: A tutorial. **European Journal of Operational Research**, v. 184, n. 2, p. 397–415, 2008. Citado nas páginas 29, 46, 74 e 102.

____. A tutorial in irregular shape packing problems. **Journal of the Operational Research Society**, v. 60, p. 93–105, 2009. Citado nas páginas 26 e 102.

BERALDI, P.; BRUNI, M. E.; CONFORTI, D. The stochastic trim-loss problem. **European Journal of Operational Research**, v. 197, n. 1, p. 42–49, 2009. Citado na página 39.

BERTSIMAS, D.; SIM, M. Robust discrete optimization and network flows. **Mathematical Programming**, v. 98, n. 1, p. 49–71, 2003. Citado na página 39.

BIAJOLI, F. L.; CHAVES, A. A.; LORENA, L. A. N. A biased random-key genetic algorithm for the two-stage capacitated facility location problem. **Expert Systems with Applications**, v. 115, p. 418–426, 2019. Citado na página 48.

BIRGE, J. R.; LOUVEAUX, F. **Introduction to stochastic programming**. New York: Springer, 1997. Citado nas páginas 38, 79, 82, 110 e 130.

BIRGIN, E. G.; SOBRAL, F. N. C. Minimizing the object dimensions in circle and sphere packing problems. **Computers & Operations Research**, v. 35, n. 7, p. 2357–2375, 2008. Citado na página 36.

BISCHOFF, E. E.; RATCLIFF, M. S. W. Issues in the development of approaches to container loading. **OMEGA**, v. 23, n. 4, p. 377–390, 1995. Citado na página 27.

BORTFELDT, A.; WÄSCHER, G. Constraints in container loading - a state-of-the-art review. **European Journal of Operational Research**, v. 229, n. 1, p. 1–20, 2013. Citado na página 27.

BURKE, E.; HELLIER, R.; KENDALL, G.; WHITWELL, G. A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. **Operations Research**, v. 54, n. 3, p. 587–601, 2006. Citado na página 35.

____. Complete and robust no-fit polygon generation for the irregular stock cutting problem. **European Journal of Operational Research**, v. 179, n. 1, p. 27–49, 2007. Citado na página 31.

BURKE, E. K.; HELLIER, R. S. R.; KENDALL, G.; WHITWELL, G. Irregular packing using the line and arc no-fit polygon. **Operations Research**, v. 58, n. 4, p. 948–970, 2010. Citado nas páginas 31 e 46.

CARRAVILLA, M. A.; RIBEIRO, C.; OLIVEIRA, J. F.; GOMES, A. M. Solving nesting problems with non-convex polygons by constraint logic programming. **International Transactions in Operational Research**, v. 10, p. 651–663, 2003. Citado nas páginas 34 e 75.

CHEN, K.; ZHUANG, J.; ZHONG, S.; ZHENG, S. Optimization method for guillotine packing of rectangular items within an irregular and defective slate. **Mathematics**, v. 8, n. 11, p. 1914, 2020. Citado nas páginas 38 e 104.

CHERNOV, N.; STOYAN, Y.; ROMANOVA, T.; PANKRATOV, A. Phi-functions for 2d objects formed by line segments and circular arcs. **Advances in Operations Research**, v. 2012, p. 346358, 2012. Citado nas páginas 32 e 46.

CHERRI, L. H. **Nesting problems**. 167 p. Tese (Doutorado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2016. Citado na página 33.

CHERRI, L. H.; CARRAVILLA, M. A.; RIBEIRO, C.; TOLEDO, F. M. B. Optimality in nesting problems: New constraint programming models and a new global constraint for non-overlap. **Operations Research Perspectives**, v. 6, p. 100125, 2019. Citado nas páginas 33 e 47.

CHERRI, L. H.; CHERRI, A. C.; CARRAVILLA, M. A.; OLIVEIRA, J. F.; TOLEDO, F. M. B.; VIANNA, A. C. G. An innovative data structure to handle the geometry of nesting problems. **International Journal of Production Research**, v. 56, n. 23, p. 7085–7102, 2018. Citado na página 30.

CHERRI, L. H.; MUNDIM, L. R.; ANDRETTA, M.; TOLEDO, F. M.; OLIVEIRA, J. F.; CARRAVILLA, M. A. Robust mixed-integer linear programming models for the irregular strip packing problem. **European Journal of Operational Research**, v. 253, n. 3, p. 570–583, 2016. Citado nas páginas 28, 34, 46, 75, 90 e 91.

CHUNG, J.; SCOTT III, D. R.; HILLMAN, D. J. Intelligent nesting system on 2-d highly irregular resources. In: TRIVEDI, M. M. (Ed.). **Applications of Artificial Intelligence VIII**. Orlando, FL, United States, 1990. v. 1293, p. 472–483. Citado na página 103.

CONFORTI, M.; CORNUÉJOLS, G.; ZAMBELLI, G. **Integer Programming**. [S.l.]: Springer Publishing Company, Incorporated, 2014. Citado na página 26.

COSTA, L. R.; ALOISE, D.; MLADENOVIC, N. Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. **Information Sciences**, v. 415-416, p. 247–253, 2017. Citado na página 47.

CRAINIC, T. G.; GOBBATO, L.; PERBOLI, G.; REI, W.; WATSON, J.-P.; WOODRUFF, D. L. Bin packing problems with uncertainty on item characteristics: An application to capacity planning in logistics. **Procedia - Social and Behavioral Sciences**, v. 111, p. 654–662, 2014. Citado na página 27.

CRAINIC, T. G.; GOBBATO, L.; PERBOLI, G.; REI, W. Logistics capacity planning: A stochastic bin packing formulation and a progressive hedging meta-heuristic. **European Journal of Operational Research**, v. 253, n. 2, p. 404–417, 2016. Citado nas páginas 39 e 74.

CRISPIN, A.; CLAY, P.; TAYLOR, G.; BAYES, T.; REEDMAN, D. Genetic algorithm coding methods for leather nesting. **Applied Intelligence**, v. 23, p. 9–20, 2005. Citado nas páginas 38, 47 e 103.

CUNINGHAME-GREEN, R. Geometry, shoemaking and the milk tray problem. **New Scientist**, v. 1677, p. 50–53, 1989. Citado nas páginas 31 e 49.

DALALAH, D.; KHRAIS, S.; BATAINEH, K. Waste minimization in irregular stock cutting. **Journal of Manufacturing Systems**, v. 33, n. 1, p. 27–40, 2014. Citado nas páginas 28, 33, 47, 65 e 103.

DERRAC, J.; GARCÍA, S.; MOLINA, D.; HERRERA, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. **Swarm and Evolutionary Computation**, v. 1, n. 1, p. 3–18, 2011. Citado na página 66.

DIGHE, R.; JAKIELA, M. J. Solving pattern nesting problems with genetic algorithms employing task decomposition and contact detection. **Evolutionary Computation**, v. 3, n. 3, p. 239–266, 1995. Citado na página 88.

DOWSLAND, K. A.; DOWSLAND, W. B.; BENNELL, J. A. Jostling for position: local improvement for irregular cutting patterns. **Journal of the Operational Research Society**, v. 49, n. 6, p. 647–658, 1998. Citado nas páginas 35 e 46.

DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research**, v. 44, n. 2, p. 145–159, 1990. Citado nas páginas 25 e 26.

ELKERAN, A. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. **European Journal of Operational Research**, v. 231, n. 3, p. 757–769, 2013. Citado nas páginas 28, 36, 46 e 75.

FISCHETTI, M.; LUZZI, I. Mixed-integer programming models for nesting problems. **Journal of Heuristics**, v. 15, p. 201–226, 2009. Citado nas páginas 34, 65 e 75.

FOWLER, R. J.; PATERSON, M. S.; TANIMOTO, S. L. Optimal packing and covering in the plane are NP-complete. **Information Processing Letters**, v. 12, n. 3, p. 133–137, 1981. Citado na página 33.

FUJITA, K.; AKAGI, S.; HIROKAWA, N. Hybrid approach for optimal nesting using a genetic algorithm and a local minimization algorithm. In: **Proceedings of the 19th Annual ASME Design Automation Conference**. Albuquerque, New Mexico, USA: [s.n.], 1993. p. 477–484. Citado nas páginas 88 e 112.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. San Francisco: Freeman, 1979. Citado nas páginas 26, 46, 75 e 105.

GOMES, A. M.; OLIVEIRA, J. F. A 2-exchange heuristic for nesting problems. **European Journal of Operational Research**, v. 141, n. 2, p. 359–370, 2002. Citado nas páginas 32, 35 e 75.

____. Solving irregular strip packing problems by hybridising simulated annealing and linear programming. **European Journal of Operational Research**, v. 171, n. 3, p. 811–829, 2006. Citado nas páginas 35, 46, 65 e 75.

GONÇALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, v. 17, n. 5, p. 487–525, 2011. Citado nas páginas 48 e 50.

GONÇALVES, J. F.; RESENDE, M. G. C. A biased random key genetic algorithm for 2D and 3D bin packing problems. **International Journal of Production Economics**, v. 145, n. 2, p. 500–510, 2013. Citado na página 48.

GUROBI OPTIMIZATION, L. **Gurobi Optimizer Reference Manual**. 2020. Disponível em: <http://www.gurobi.com>. Citado nas páginas 86 e 111.

HAN, G. C.; NA, S. J. A new multi-stage layout approach for optimal nesting of 2-dimensional patterns with boundary constraints and internal defects. **Transactions of the Korean Society of Mechanical Engineers**, v. 18, n. 12, p. 3236–3245, 1994. Citado na página 103.

HAN, G.-C.; NA, S.-J. Two-stage approach for nesting in two-dimensional cutting problems using neural network and simulated annealing. **Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture**, v. 210, n. 6, p. 509–519, 1996. Citado nas páginas 62 e 88.

HANSEN, P.; MLADENOVIĆ, N.; PÉREZ, J. M. Variable neighbourhood search: methods and applications. **Annals of Operations Research**, v. 175, n. 1, p. 367–407, 2010. Citado nas páginas 48, 62 e 83.

HEISTERMANN, J.; LENGAUER, T. The nesting problem in the leather manufacturing industry. **Annals of Operations Research**, v. 57, p. 147–173, 1995. Citado nas páginas 37 e 103.

HERRÁN, A.; COLMENAR, J. M.; DUARTE, A. A variable neighborhood search approach for the hamiltonian p-median problem. **Applied Soft Computing**, v. 80, p. 603–616, 2019. Citado na página 48.

HIFI, M.; WU, L. Lagrangian heuristic-based neighbourhood search for the multiple-choice multi-dimensional knapsack problem. **Engineering Optimization**, v. 47, n. 12, p. 1619–1636, 2015. Citado na página 48.

HOPPER, E. **Mathematical models and heuristic methods for nesting problems**. 80 p. Tese (Doutorado) — School of Engineering, University of Wales, Cardiff, 2000. Citado nas páginas 88 e 112.

HOTTUNG, A.; TIERNEY, K. A biased random-key genetic algorithm for the container pre-marshalling problem. **Computers & Operations Research**, v. 75, p. 83–102, 2016. Citado na página 48.

IMAMICHI, T.; YAGIURA, M.; NAGAMOCHI, H. An iterated local search algorithm based on nonlinear programming for the irregular strip packing problem. **Discrete Optimization**, v. 6, n. 4, p. 345–361, 2009. Citado na página 36.

JAKOBS, S. On genetic algorithms for the packing of polygons. **European Journal of Operational Research**, v. 88, n. 1, p. 165–181, 1996. Citado nas páginas 35, 75 e 88.

JUNQUEIRA, L. **Modelos de programação matemática para problemas de carregamento de caixas dentro de contêineres**. 134 p. Dissertação (Mestrado em Engenharia de Produção) — Universidade Federal de São Carlos, São Carlos, SP, Brasil, 2009. Citado nas páginas 82 e 109.

KALL, P.; WALLACE, S. **Stochastic programming**. New York: Wiley, 1994. Citado nas páginas 38 e 39.

KALLRATH, J. Cutting circles and polygons from area-minimizing rectangles. **Journal of Global Optimization**, v. 43, n. 2, p. 299–328, 2009. Citado nas páginas 36 e 46.

KOTHARI, C. R. **Research Methodology: Methods and Techniques**. New Delhi: New Age International, 2004. Citado na página 28.

KUMAR, G. N.; BANGI, M. An extension to winding number and point-in-polygon algorithm. **IFAC-PapersOnLine**, v. 51, n. 1, p. 548–553, 2018. 5th IFAC Conference on Advances in Control and Optimization of Dynamical Systems ACODS 2018. Citado na página 30.

LALLA-RUIZ, E.; VOSS, S. A biased random-key genetic algorithm for the multiple knapsack assignment problem. In: DHAENENS, C.; JOURDAN, L.; MARMION, M.-E. (Ed.). **Learning and Intelligent Optimization**. Cham: Springer International Publishing, 2015. p. 218–222. Citado na página 48.

LEÃO, A. A. S.; TOLEDO, F. M. B.; OLIVEIRA, J. F.; CARRAVILLA, M. A. A semi-continuous mip model for the irregular strip packing problem. **International Journal of Production Research**, v. 54, n. 3, p. 712–721, 2016. Citado nas páginas 34 e 75.

LEÃO, A. A. S.; TOLEDO, F. M. B.; OLIVEIRA, J. F.; CARRAVILLA, M. A.; ALVAREZ-VALDES, R. Irregular packing problems: A review of mathematical models. **European Journal of Operational Research**, v. 282, n. 3, p. 803–822, 2020. Citado nas páginas 28, 35, 76 e 88.

LEE, W.-C.; MA, H.; CHENG, B.-W. A heuristic for nesting problems of irregular shapes. **Computer-Aided Design**, v. 40, n. 5, p. 625–633, 2008. Citado nas páginas 38 e 104.

LI, J.-Q.; PAN, Q.-K.; WANG, F.-T. A hybrid variable neighborhood search for solving the hybrid flow shop scheduling problem. **Applied Soft Computing**, v. 24, p. 63–77, 2014. Citado na página 48.

LI, Z.; MILENKOVIC, V. Compaction and separation algorithms for non-convex polygons and their applications. **European Journal of Operational Research**, v. 84, n. 3, p. 539–561, 1995. Citado na página 34.

LIU, W.; DENG, T.; LI, J. Product packing and stacking under uncertainty: A robust approach. **European Journal of Operational Research**, v. 277, n. 3, p. 903–917, 2019. Citado na página 39.

LÓPEZ-CAMACHO, E.; TERASHIMA-MARÍN, H.; ROSS, P.; OCHOA, G. A unified hyper-heuristic framework for solving bin packing problems. **Expert Systems with Applications**, v. 41, n. 15, p. 6876–6889, 2014. Citado na página 37.

LOPEZ-IBANEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. **Operations Research Perspectives**, v. 3, p. 43–58, 2016. Citado na página 60.

MA, Z.; KWON, R. H.; LEE, C.-G. A stochastic programming winner determination model for truckload procurement under shipment uncertainty. **Transportation Research Part E: Logistics and Transportation Review**, v. 46, p. 49–60, 2010. Citado nas páginas 88 e 112.

MAHADEVAN, A. **Optimization in Computer-aided Pattern Packing**. 203 p. Tese (Doutorado) — North Carolina State University, Raleigh, NC, USA, 1984. Citado na página 30.

MARTINS, T. C.; TSUZUKI, M. S. G. Simulated annealing applied to the irregular rotational placement of shapes over containers with fixed dimensions. **Expert Systems with Applications**, v. 37, n. 3, p. 1955–1972, 2010. Citado nas páginas 28, 33, 47, 65 e 102.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, p. 1097–1100, 1997. Citado nas páginas 48 e 55.

MONACI, M.; PFERSCHY, U.; SERAFINI, P. Exact solution of the robust knapsack problem. **Computers & Operations Research**, v. 40, n. 11, p. 2625–2631, 2013. Citado na página 39.

MUNDIM, L. R. **Uma abordagem heurística para o corte de itens irregulares em múltiplos recipientes**. 123 p. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2015. Citado nas páginas 29 e 31.

MUNDIM, L. R. **Mathematical models and heuristic methods for nesting problems**. 167 p. Tese (Doutorado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2017. Citado nas páginas 28, 38, 41, 76 e 104.

MUNDIM, L. R.; ANDRETTA, M. Problema de corte de itens irregulares na fabricação de luvas de couro. In: **XLVI Simpósio Brasileiro de Pesquisa Operacional**. Salvador-BA, Brasil: [s.n.], 2014. p. 1–12. Citado na página 104.

MUNDIM, L. R.; ANDRETTA, M.; CARRAVILLA, M. A.; OLIVEIRA, J. F. A general heuristic for two-dimensional nesting problems with limited-size containers. **International Journal of Production Research**, v. 56, n. 1-2, p. 709–732, 2018. Citado nas páginas 28, 33, 37, 38, 40, 41, 46, 47, 58, 61, 62, 65, 66, 68 e 103.

MUNDIM, L. R.; ANDRETTA, M.; QUEIROZ, T. A. A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. **Expert Systems with Applications**, v. 81, p. 358–371, 2017. Citado nas páginas 36, 37, 46, 48, 58 e 75.

MUNDIM, L. R.; QUEIROZ, T. A. de. A hybrid heuristic for the 0–1 knapsack problem with items of irregular shape. In: **2012 XXXVIII Conferencia Latinoamericana En Informatica (CLEI)**. [S.l.: s.n.], 2012. p. 1–6. Citado nas páginas 33 e 103.

NASCIMENTO, O. X.; QUEIROZ, T. A.; JUNQUEIRA, L. Practical constraints in the container loading problem: Comprehensive formulations and exact algorithm. **Computers & Operations Research**, v. 128, p. 105186, 2021. Citado na página 27.

OLIVEIRA, J. F. **Problemas de posicionamento de figuras irregulares: uma perspectiva de optimização**. 282 p. Tese (Doutorado em Engenharia Electrotécnia e de Computadores) — Faculdade de Engenharia, Universidade do Porto, Porto, Portugal, 1995. Citado nas páginas 27 e 33.

OLIVEIRA, J. F.; FERREIRA, J. A. S. Algorithms for nesting problems. In: ____. **Applied Simulated Annealing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993. p. 255–273. Citado nas páginas 30 e 46.

OLIVEIRA, J. F.; GOMES, A. M.; FERREIRA, J. S. TOPOS – a new constructive algorithm for nesting problems. **OR-Spektrum**, v. 22, n. 2, p. 263–284, 2000. Citado nas páginas 28, 35, 62, 88 e 112.

PAN, F.; NAGI, R. Robust supply chain design under uncertain demand in agile manufacturing. **Computers & Operations Research**, v. 37, n. 4, p. 668–683, 2010. Citado na página 88.

PEKEL, E.; KARA, S. S. Solving fuzzy capacitated location routing problem using hybrid variable neighborhood search and evolutionary local search. **Applied Soft Computing**, v. 83, p. 105665, 2019. Citado na página 48.

PERALTA, J.; ANDRETTA, M.; OLIVEIRA, J. F. Solving irregular strip packing problems with free rotations using separation lines. **Pesquisa Operacional**, v. 38, n. 2, p. 195–214, 2018. Citado nas páginas 30, 34 e 46.

PERBOLI, G.; TADEI, R.; BALDI, M. M. The stochastic generalized bin packing problem. **Discrete Applied Mathematics**, v. 160, n. 7, p. 1291–1297, 2012. Citado na página 39.

PINHEIRO, P. R.; AMARO JR., B.; SARAIVA, R. D. A random-key genetic algorithm for solving the nesting problem. **International Journal of Computer Integrated Manufacturing**, v. 29, n. 11, p. 1159–1165, 2016. Citado nas páginas 36 e 48.

PINTO, T. M. P.; ALVES, C. M. M.; CARVALHO, J. M. V.; BRÁS, P. A. F. Heuristic methods for the leather nesting problem in the automotive industry. **International Journal of Business Excellence**, v. 9, n. 3, p. 332–347, 2016. Citado nas páginas 38, 46, 47 e 104.

POLO, J. M. P. **Resolução de problemas de empacotamento de itens irregulares usando técnicas de programação não-linear**. 103 p. Tese (Doutorado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2018. Citado na página 33.

QIU, Y.; WANG, L.; XU, X.; FANG, X.; PARDALOS, P. M. A variable neighborhood search heuristic algorithm for production routing problems. **Applied Soft Computing**, v. 66, p. 311–318, 2018. Citado na página 48.

RANGE, T. M.; KOZLOWSKI, D.; PETERSEN, N. C. A shortest-path-based approach for the stochastic knapsack problem with non-decreasing expected overfilling costs. **Computers & Operations Research**, v. 97, p. 111–124, 2018. Citado na página 40.

REIJNTJES, I. E. **Solving a nesting problem using the collision free region**. 46 p. Dissertação (Master in Mathematics and Computer Science) — Eindhoven University of Technology, Eindhoven, Netherlands, 2016. Citado na página 104.

ROCHA, P.; GOMES, A. M.; RODRIGUES, R.; TOLEDO, F. M. B.; ANDRETTA, M. Constraint aggregation in non-linear programming models for nesting problems. In: FONSECA, R. J.; WEBER, G.-W.; TELHADA, J. (Ed.). **Computational Management Science**. Cham: Springer International Publishing, 2016. p. 175–180. Citado na página 30.

ROCHA, P.; RODRIGUES, R.; GOMES, A. M.; TOLEDO, F. M. B.; ANDRETTA, M. Circle covering representation for nesting problems with continuous rotations. **IFAC Proceedings Volumes**, v. 47, n. 3, p. 5235–5240, 2014. Citado na página 30.

RODRIGUES, M. O.; TOLEDO, F. M. A clique covering mip model for the irregular strip packing problem. **Computers & Operations Research**, v. 87, p. 221–234, 2017. Citado nas páginas 34, 76, 88, 90, 91 e 112.

ROMÁN, A. E. **An Evolutionary Algorithm for Solving the Two-Dimensional Irregular Shape Packing Problem Combined with the Knapsack Problem**. 116 p. Dissertação (Mestrado em Artificial Intelligence) — Polytechnic University of Catalonia, Barcelona, Spain, 2020. Citado nas páginas 28 e 103.

ROODERKERK, R. P.; HEERDE, H. J. Robust optimization of the 0-1 knapsack problem: Balancing risk and return in assortment optimization. **European Journal of Operational Research**, v. 250, n. 3, p. 842–854, 2016. Citado na página 39.

RUIZ, E.; SOTO-MENDOZA, V.; BARBOSA, A. E. R.; REYES, R. Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. **Computers & Industrial Engineering**, v. 133, p. 207–219, 2019. Citado na página 48.

SANTOS, L. F. M.; IWAYAMA, R. S.; CAVALCANTI, L. B.; TURI, L. M.; MORAIS, F. E. de S.; MORMILHO, G.; CUNHA, C. B. A variable neighborhood search algorithm for the bin packing problem with compatible categories. **Expert Systems with Applications**, v. 124, p. 209–225, 2019. Citado na página 48.

SARPER, H.; JAKSIC, N. I. Simulation of the stochastic one-dimensional cutting stock problem to minimize the total inventory cost. **Procedia Manufacturing**, v. 38, p. 916–923, 2019. Citado na página 74.

SATO, A. K.; MARTINS, T. C.; GOMES, A. M.; TSUZUKI, M. S. G. Raster penetration map applied to the irregular packing problem. **European Journal of Operational Research**, v. 279, n. 2, p. 657–671, 2019. Citado nas páginas 30 e 36.

SATO, A. K.; MARTINS, T. C.; TSUZUKI, M. S. G. An algorithm for the strip packing problem using collision free region and exact fitting placement. **Computer-Aided Design**, v. 44, n. 8, p. 766–777, 2012. Citado na página 30.

SCHEITHAUER, G. **Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods**. [S.l.]: Springer, Cham, 2017. v. 263. Citado nas páginas 25 e 45.

SCHEITHAUER, G.; TERNO, J. Modeling of packing problems. **Optimization**, v. 28, n. 1, p. 63–84, 1993. Citado nas páginas 33, 46 e 102.

SEGENREICH, S. A.; BRAGA, L. M. Optimal nesting of general plane figures: a monte carlo heuristical approach. **Computers and Graphics**, v. 10, p. 229–237, 1986. Citado na página 30.

SILVA, R. A. O. K. **Empacotamento de itens irregulares considerando balanceamento da carga**. 115 p. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) — Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, SP, Brasil, 2017. Citado na página 33.

SILVEIRA, T. **Problemas de Empacotamento com Itens Irregulares: Heurísticas e Avaliação de Construtores de NFP**. 112 p. Dissertação (Mestrado em Ciências da Computação) — Instituto de Computação, Universidade de Campinas, Campinas, SP, Brasil, 2013. Citado nas páginas 33, 48 e 103.

SONG, X.; BENNELL, J. A. Column generation and sequential heuristic procedure. **Journal of the Operational Research Society**, v. 65, n. 7, p. 1037–1052, 2014. Citado na página 37.

____. Column generation and sequential heuristic procedure for solving an irregular shape cutting stock problem. **Journal of the Operational Research Society**, v. 65, n. 7, p. 1037–1052, 2014. Citado na página 46.

SOUZA QUEIROZ, L. R.; ANDRETTA, M. Modelo de programação estocástica para um problema de corte de itens irregulares. In: **LII Simpósio Brasileiro de Pesquisa Operacional**. João Pessoa, PB, Brasil: [s.n.], 2020. p. 1–12. Citado nas páginas 42 e 73.

____. Two effective methods for the irregular knapsack problem. **Applied Soft Computing**, v. 95, p. 106485, 2020. Citado nas páginas 41, 45, 83, 103 e 112.

____. Problema da mochila com itens irregulares e incerteza nos defeitos da placa. In: **LIII Simpósio Brasileiro de Pesquisa Operacional**. João Pessoa, PB, Brasil: [s.n.], 2021. p. 1–11. Citado nas páginas 43 e 101.

____. A branch-and-cut algorithm for the irregular strip packing problem with uncertain demands. **International Transactions in Operational Research**, n/a, 2022. Citado nas páginas 42 e 73.

SOUZA QUEIROZ, L. R.; MUNDIM, L. R.; ANDRETTA, M. Genetic algorithm for the knapsack problem with irregular shaped items. In: **2018 XLIV Latin American Computer Conference (CLEI)**. [S.l.: s.n.], 2018. p. 192–199. Citado nas páginas 41 e 45.

SOYSTER, A. L. Convex programming with set-inclusive constraints and applications to inexact linear programming. **Operations Research**, v. 21, n. 5, p. 1154–1157, 1973. Citado na página 74.

STOYAN, Y.; PANKRATOV, A.; ROMANOVA, T. Quasi-phi-functions and optimal packing of ellipses. **Journal of Global Optimization**, v. 65, p. 283–307, 2016. Citado na página 32.

STOYAN, Y.; ROMANOVA, T.; PANKRATOV, A.; CHUGAY, A. Optimized object packings using quasi-phi-functions. In: ____. **Optimized Packings with Applications**. Cham: Springer International Publishing, 2015. p. 265–293. Citado na página 32.

STOYAN, Y. G.; PATSUK, V. A method of optimal lattice packing of congruent oriented polygons in the plane. **European Journal of Operational Research**, v. 124, n. 1, p. 204–216, 2000. Citado na página 36.

STOYAN, Y. G.; SCHEITHAUER, G.; GIL, N.; ROMANOVA, T. Phi-functions for complex 2D-objects. **4OR quarterly journal of the Belgian, French and Italian Operations Research Societies**, v. 2, p. 69–84, 2004. Citado na página 31.

STOYAN, Y. G.; TERNO, J.; SCHEITHAUER, G.; GIL, N.; ROMANOVA, T. Phi-functions for primary 2D-objects. **Studia Informatica Universalis**, v. 1, p. 1–32, 2001. Citado na página 31.

SYKORA, A. M. **Nesting Problems: Exact and Heuristic Algorithms**. 192 p. Tese (Doutorado) — Faculty of Mathematics, University of Valencia, Valencia, Spain, 2013. Citado na página 33.

SYKORA, A. M.; ALVAREZ-VALDES, R.; BENNELL, J. A.; RUIZ, R.; TAMARIT, J. M. Matheuristics for the irregular bin packing problem with free rotations. **European Journal of Operational Research**, v. 258, n. 2, p. 440–455, 2017. Citado na página 37.

TALBI, E.-G. **Metaheuristics: From Design to Implementation**. New Jersey: John Wiley & Sons, 2009. Citado na página 26.

TAY, F. E. H.; CHONG, T. Y.; LEE, F. C. Pattern nesting on irregular-shaped stock using genetic algorithms. **Engineering Applications of Artificial Intelligence**, v. 15, n. 6, p. 551–558, 2002. Citado na página 103.

TERASHIMA-MARÍN, H.; ROSS, P.; FARÍAS-ZÁRATE, C. J.; LÓPEZ-CAMACHO, E.; VALENZUELA-RENDÓN, M. Generalized hyper-heuristics for solving 2d regular and irregular packing problems. **Annals of Operations Research**, v. 179, p. 369–392, 2010. Citado na página 37.

TOLEDO, F. M. B.; CARRAVILLA, M. A.; RIBEIRO, C.; OLIVEIRA, J. F.; GOMES, A. M. The dotted-board model: a new mip model for nesting irregular shapes. **International Journal of Production Economics**, v. 145, n. 2, p. 478–487, 2013. Citado nas páginas 30, 31, 34, 46, 62, 74, 75, 76, 78, 80, 88, 105, 106 e 112.

TOSO, R.; RESENDE, M. A C++ application programming interface for biased random-key genetic algorithms. **Optimization Methods and Software**, v. 30, n. 1, p. 81–93, 2015. Citado na página 51.

VALLE, A. M. D.; QUEIROZ, T. A.; MIYAZAWA, F. K.; XAVIER, E. C. Heuristics for two-dimensional knapsack and cutting stock problems with items of irregular shape. **Expert Systems with Applications**, v. 39, n. 16, p. 12589–12598, 2012. Citado nas páginas 28, 33, 37, 46, 47, 61, 62, 65, 102 e 103.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, n. 3, p. 1109–1130, 2007. Citado nas páginas 25, 26, 27, 28, 46, 74 e 102.

WONG, W. K.; WANG, X. X.; MOK, P. Y.; LEUNG, S. Y. S.; KWONG, C. K. Solving the two-dimensional irregular objects allocation problems by using a two-stage packing approach. **Expert Systems with Applications**, v. 36, n. 2, Part 2, p. 3489–3496, 2009. Citado na página 36.

XU, Y.-X. An efficient heuristic approach for irregular cutting stock problem in ship building industry. **Mathematical Problems in Engineering**, v. 2016, p. 8703782, 2016. Citado na página 37.

YUPING, Z.; SHOUWEI, J.; CHUNLI, Z. A very fast simulated re-annealing algorithm for the leather nesting problem. **International Journal of Advanced Manufacturing Technology**, v. 25, n. 11, p. 1113–1118, 2005. Citado na página 103.

ZENG, Z.-Z.; YU, X.-G.; HE, K.; FU, Z.-H. Adaptive tabu search and variable neighborhood descent for packing unequal circles into a square. **Applied Soft Computing**, v. 65, p. 196–213, 2018. Citado na página 48.


ICMC
USP
SÃO CARLOS