UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

O Problema do Empacotamento de Itens Irregulares em *Bins*

Diego Yoshihiro Hono

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

ICMC USP SÃO CARLOS

Diego Yoshihiro Hono

O Problema do Empacotamento de Itens Irregulares em *Bins*

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

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Franklina M. B. de Toledo
Coorientador: Profa. Dra. Aline A. S. Leão

USP – São Carlos
Junho de 2024

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

Hono, Diego Yoshihiro
H744p O Problema do Empacotamento de Itens Irregulares em Bins / Diego Yoshihiro Hono; orientadora Franklina Maria Bragion de Toledo; coorientadora Aline Aparecida de Souza Leão. -- São Carlos, 2024.
105 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. Empacotamento em bins. 2. Peças irregulares. 3. Programação matemática. 4. Método heurístico. 5. Algoritmo genético. I. Toledo, Franklina Maria Bragion de, orient. II. Leão, Aline Aparecida de Souza , coorient. III. 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

Diego Yoshihiro Hono

Irregular bin packing problem

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Franklina M. B. de Toledo
Co-advisor: Profa. Dra. Aline A. S. Leão

USP – São Carlos
June 2024

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001, do auxílio do Programa de Excelência Acadêmica (Proex) — CAPES (88887.685114/2022-00), da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) (2013/07375-0 (CEPID-CeMEAI)). Os autores também agradecem ao Laboratório de Otimização do ICMC/USP (LOt).

RESUMO

HONO, D. Y. **O Problema do Empacotamento de Itens Irregulares em *Bins***. 2024. 105 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Na indústria de confecção de roupas, para manter a qualidade de um produto e agilizar sua produção, todas as partes que o compõem devem ser cortadas do mesmo tecido, por exemplo, as mangas, a frente e as costas de uma camiseta devem ser cortadas sempre em conjunto. Além disso, a aquisição do tecido representa um custo elevado para a indústria de vestuário, portanto, a redução do desperdício de tecido tem impactos econômicos e ambientais positivos. O processo de corte começa dispondo o tecido sobre uma mesa de corte com largura fixa e comprimento que pode ser fixado dentro de uma dada faixa. Um plano de corte deve ser definido em seguida, ou seja, os itens precisam ser alocados neste tecido retangular (*bin*). O objetivo é maximizar o aproveitamento da matéria-prima ao gerar planos de corte que atendam à demanda de produtos finais (por exemplo, camisa, camiseta ou calça). Do ponto de vista de otimização, este problema pode ser tratado como um problema de empacotamento bidimensional em *bins*, pois demandas grandes exigem com que mais de um plano de corte seja elaborado e as dimensões de cada *bin* são limitadas devido ao tamanho das mesas de corte. Além disso, também é permitida uma pequena redução no comprimento do *bin* para reduzir o desperdício de matéria-prima no caso em que não é possível empacotar mais produtos. Neste trabalho, é apresentado um modelo de programação linear inteira-mista para representar o problema e uma heurística *biased random key genetic algorithm* (BRKGA) para resolver o problema. Experimentos computacionais mostraram que as soluções obtidas ao representar as instâncias pelo modelo proposto e resolvê-las utilizando o solver Gurobi (v9.5.1), são melhores em relação à qualidade, sendo 21,2% melhores em média, porém seu custo computacional é significativamente maior, o que torna a heurística uma opção viável em casos em que as instâncias são maiores.

Palavras-chave: Empacotamento em *bins*; peças irregulares; programação matemática; modelagem; método heurístico; algoritmo genético.

ABSTRACT

HONO, D. Y. **Irregular bin packing problem**. 2024. 105 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

In the clothing industry, in order to maintain the quality of a product and speed up its production, all the parts that make it up must be cut from the same fabric, for example, the sleeves, front and back of a T-shirt must always be cut together. In addition, the purchase of fabric represents a high cost for the clothing industry, so reducing fabric waste has positive economic and environmental impacts. The cutting process begins by laying out the fabric on a cutting table with a fixed width and a length that can be set within a given range. A cutting plan must then be defined, i.e. the items need to be allocated to this rectangular fabric (bin). The aim is to maximize the use of raw materials by generating cutting plans that meet the demand for end products (e.g. shirt, t-shirt or pants). From an optimization point of view, this problem can be treated as a two-dimensional bin packing problem, because large demands require more than one cutting plan to be drawn up and the dimensions of each bin are limited due to the size of the cutting tables. In addition, a small reduction in the length of the bin is also allowed in order to reduce raw material waste in the case where it is not possible to package more products. In this work, a mixed integer linear programming model is presented to represent the problem and a heuristic biased random key genetic algorithm (BRKGA) to solve the problem. Computational experiments have shown that the solutions obtained by representing the instances using the proposed model and solving them using the Gurobi solver (v9.5.1), are better in terms of quality, being 21.2% better on average, but their computational cost is significantly higher, which makes the heuristic a viable option in cases where the instances are larger.

Keywords: Bin packing, irregular shapes, mathematical programming, modeling, heuristic method, genetic algorithm.

LISTA DE ILUSTRAÇÕES

LISTA DE ALGORITMOS

LISTA DE TABELAS

SUMÁRIO

CAPÍTULO

1

INTRODUÇÃO

Segundo a Associação Brasileira da Indústria Têxtil e de Confecção (ABIT (2023)), o setor faturou R$ 190 bilhões em 2021 e empregou 1,34 milhões de pessoas de forma direta em 2022 no Brasil. Considerando os empregos indiretos, esse número pode chegar a 8 milhões. Sendo assim, o setor foi responsável por aproximadamente 6% do faturamento e 19,5% dos empregados alocados na produção industrial no ano de 2021. Os dados mostram sua relevância para a economia brasileira e apontam para a importância do estudo de técnicas que aumentem sua produtividade.

Além disso, segundo uma reportagem feita pelo National Geographic (2024), o excesso de resíduos proveniente das indústrias têxteis é um problema no mundo. Por exemplo, nos países Chile e Gana, os retalhos de tecidos gerados após o corte dos produtos são descartados ao ar livre devido ao grande volume. Assim, o bom funcionamento deste setor não tem apenas um impacto social e econômico, mas também um impacto ambiental.

Uma das formas de aumentar a produtividade do setor e reduzir seu impacto ambiental, é contribuir para reduzir o uso de matéria-prima. Segundo Alsamarah, Younes e Yousef (2022), o processo de corte é um dos principais responsáveis pela agregação de valor aos produtos. Nesta direção, a definição de bons arranjos das peças para o corte da matéria-prima se torna fundamental para maximizar o aproveitamento do material e agilizar o processo de produção, desonerando os custos de produção e minimizando os resíduos descartados.

Na literatura, o problema de definição de arranjos para o corte é conhecido como problema de corte e empacotamento, que devido à sua importância prática e ao desafio teórico associado é estudado há algumas décadas (Gilmore e Gomory (1961), Gilmore e Gomory (1963), Dowsland e Dowsland (1995), Bennell e Oliveira (2009), Leão *et al.* (2020)). Em linhas gerais, o problema consiste em encontrar um arranjo para peças menores dentro de um objeto maior. Para que este arranjo (plano de corte) seja considerado viável, as peças devem estar inteiramente dispostas no objeto e não podem se sobrepor. Ao elaborar um plano de corte, vários objetivos

podem ser considerados, no entanto, um muito frequente é maximizar o aproveitamento da matéria-prima, o que contribui para a minimização dos custos de produção e da geração de resíduos. O objetivo considerado e as características do objeto e das peças a serem cortadas levam a diferentes classificações dos problemas de corte e empacotamento (ver, por exemplo, Dyckhoff e Finke (1992) e Wascher, Haußner e Schumann (2007)).

Como descrito por Alsamarah, Younes e Yousef (2022), de forma geral, o processo de produção em confecções pode ser dividido em quatro etapas principais: concepção dos produtos (incluindo a modelagem), corte, costura e embalagem. Mais especificamente, na etapa de corte, o rolo de tecido é disposto em uma mesa de corte criando camadas de tecido (enfesto do tecido). Como resultado, tem-se um objeto retangular para o qual um plano de corte deve ser definido. A partir deste objeto são cortadas as peças que compõem as roupas, frequentemente peças irregulares, ou seja, não podem ser trivialmente representadas por retângulos, círculos, e outros polígonos regulares.

Neste trabalho, será o problema de corte inspirado na confecção de roupas. Mais detalhadamente, as roupas (produtos finais) são compostas por diferentes peças (itens). Por exemplo, uma camiseta é composta por duas mangas, pela frente e pelas costas, logo para sua produção, estas quatro peças devem ser cortadas a partir de um mesmo objeto. Esta questão é importante, em especial, por dois motivos: a) permitir que ao término do corte de um objeto, o processo de costura das roupas possa ser iniciado; e b) assegurar a padronização das roupas, evitando possíveis diferenças de tons do tecido entre peças de um mesmo produto. Dessa forma, o objetivo é atender as demandas de cada produto final ao mesmo tempo que se minimiza a perda de matéria-prima. O número de planos de corte elaborados depende da demanda. Vale ressaltar que as dimensões do objeto de corte são limitadas pelo tamanho da mesa de corte. Entretanto, é permitida uma pequena redução no comprimento do *bin* para reduzir o desperdício de matéria-prima no caso em que não é possível empacotar mais produtos. Embora este seja um problema relevante para a indústria e desafiador do ponto de vista teórico, apenas M'Hallah e Bouziri (2016) abordaram o problema na literatura.

1.1 Questões de pesquisa

Esta pesquisa tem por objetivo responder três questões de pesquisa:

Q1: *É relevante considerar o corte de produtos finais em lugar do corte dos itens individualmente?*

Segundo M'Hallah e Bouziri (2016), os itens que compõem um mesmo produto final devem ser empacotados em conjunto para assegurar uma padronização na tonalidade do produto final. Entretanto, supondo que isso não é necessário, será feito um estudo para verificar a quantidade de produtos finais obtidos ao término do corte de um *bin*.

Q2: *É possível resolver instâncias até qual dimensão utilizando modelagem matemática e softwares de otimização?*

Na literatura, métodos heurísticos são mais explorados devido ao elevado custo computacional dos métodos exatos. Desta forma, este trabalho estudará a possibilidade de resolver o problema utilizando um modelo de programação linear inteira-mista.

Q3: *É possível resolver o problema utilizando BRKGA e obter soluções de alta qualidade?*

A heurística *Biased Random Key Genetic Algorithm* (BRKGA) é amplamente utilizada para resolver o problema de empacotamento de itens irregulares bidimensionais devido ao seu bom desempenho e também a sua flexibilidade de implementação. Por exemplo, Mundim, Andretta e Queiroz (2017), Souza Queiroz e Andretta (2020) e Lu, Hu e Ng (2023) utilizam a heurística em seus trabalhos. Como o problema é da classe NP-difícil de acordo com Fowler, Paterson e Tanimoto (1981), o custo computacional ao resolvê-lo utilizando um modelo matemático é atualmente elevado e, portanto, será implementado um BRKGA e comparados seus resultados com os obtidos pelo método exato.

1.2 Estrutura do texto

Este texto está organizado nos seguintes capítulos. No Capítulo 2, é apresentada uma revisão bibliográfica sobre o tema estudado. O problema estudado, modelagem matemática e experimentos computacionais envolvendo o modelo são descritos no Capítulo 3. No Capítulo 4, é proposta uma heurística para abordar o problema e seus resultados são comparados com os obtidos pela resolução do modelo pelo Gurobi. Por fim, no Capítulo 5, são apresentadas as conclusões deste trabalho.

CAPÍTULO

2

REVISÃO BIBLIOGRÁFICA

Um problema de corte e empacotamento (PCE) pode ser definido em linhas gerais como a tarefa de alocar peças (itens menores) dentro de um ou mais recipientes (objetos de cortes) de modo que as peças estejam totalmente contidas dentro do(s) recipiente(s) e sem que ocorra sobreposição entre elas.

Como destacado na literatura, os PCEs são muito relevantes e estão presentes em diferentes contextos. Dyckhoff e Finke (1992) e Wascher, Haußner e Schumann (2007) apontam que os PCEs podem ser classificados de acordo com algumas de suas características como: o número de dimensões dos itens e do recipiente (unidimensional, bidimensional ou tridimensional), tipo de itens empacotados (regulares ou irregulares) e objetivo do empacotamento (maximizar o uso do objeto de corte ou minimizar o número de recipientes para realizar o empacotamento).

Dentre os muitos problemas de corte e empacotamento, este trabalho estuda o problema de empacotamento de itens irregulares bidimensionais que também é conhecido na literatura como problema de *nesting*. Como mostraram Fowler, Paterson e Tanimoto (1981), este problema é classificado como NP-difícil.

Os problemas de corte com itens irregulares têm aplicações em diferentes indústrias como a têxtil e de confecção (p.e. Heckmann e Lengauer (1995) e M'Hallah e Bouziri (2016)), a calçadista (p.e. Heistermann e Lengauer (1995)) e a metalúrgica (p.e. Plankovskyy *et al.* (2020)). Entre suas aplicações mais recentes, são citados os setores de aviação e construção civil. Qin *et al.* (2018), Li *et al.* (2019) e Luo e Rao (2023) estudaram a alocação de aeronaves em hangares de manutenção de forma a maximizar o uso do espaço. Enquanto Kim e Lee (2021) modelaram o planejamento de leiautes de edifícios em um terreno como um problema de cortes.

Mais especificamente, será estudado o problema de *nesting* aplicado à indústria têxtil. Nesta indústria, a produção de uma peça de roupa demanda o corte das partes que a compõem. Por exemplo, uma camiseta é composta pela frente, costas e duas mangas. Posteriormente, estas partes são costuradas e finalizadas para obter um produto final. As partes são recortadas de um

tecido retangular com altura fixa e comprimento limite igual ao comprimento da mesa de corte, podendo ser diminuído caso não seja possível recortar mais partes. Vários autores abordam o problema no âmbito de confecções, por exemplo, Heckmann e Lengauer (1995), Takahara, Kusumoto e Miyamoto (2003), Amaro-Junior, Pinheiro e Saraiva (2013), M'Hallah e Bouziri (2016), Tsao, Vu e Liao (2020), Alsamarah, Younes e Yousef (2022) e Tsao, Delicia e Vu (2022).

A revisão bibliográfica estará restrita aos trabalhos que propõem modelos e métodos para resolver o problema de *nesting* que consideram objetos de corte retangulares. Devido ao desafio geométrico que este problema possui, também serão apresentadas as principais ferramentas geométricas utilizadas para tratá-lo. Em seguida, na Seção 2.2, são reportados os principais modelos de programação linear inteira-mista propostos para o problema. Na Seção 2.3, é apresentado um levantamento das principais heurísticas utilizadas para resolvê-lo.

2.1 Ferramentas geométricas

Em problemas de empacotamento de peças irregulares, é sempre necessário respeitar duas restrições: i) todos os itens devem estar totalmente contidos no objeto de corte e ii) não pode haver sobreposição entre eles, pois em ambos os casos, seriam obtidos itens defeituosos após o corte. Na Figura 1a, é ilustrado um exemplo em que um item está parcialmente fora do objeto de corte, enquanto, na Figura 1b, um exemplo de sobreposição entre dois itens é apresentado.

Há várias ferramentas geométricas utilizadas para assegurar que essas restrições sejam respeitadas (Bennell e Oliveira (2008)). A sobreposição entre dois itens pode ser verificada utilizando trigonometria direta (ou *d-functions*), *no-fit polygons*, método *raster* ou *phi-functions*. A seguir, essas ferramentas são descritas com exceção da *phi-function*, pois ela é normalmente utilizada para tratar problemas de *nesting* com rotações contínuas dos itens, aparecendo em modelos de programação não-linear, o que não é tratado no problema estudado neste trabalho.

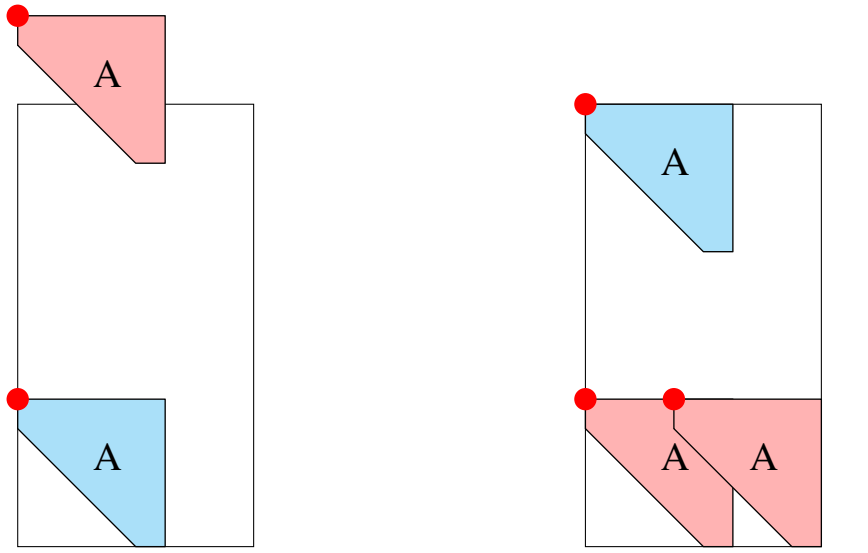
Vale ressaltar que uma revisão sobre esse tema está disponível em Bennell e Oliveira (2008). Para evitar que um item fique parcialmente fora do objeto de corte, uma das ferramentas geométricas mais utilizada é o *inner-fit polygon*, porém também é possível verificar se o item está fora do objeto de corte utilizando as técnicas de verificação de sobreposição, pois basta verificar se existe sobreposição entre o item e a região complementar do objeto de corte.

2.1.1 Trigonometria direta ou D-function

Considerando o plano cartesiano, dado um ponto $P = (p_x, p_y)$ e uma aresta orientada $\overline{AB}$, em que $A = (a_x, a_y)$ e $B = (b_x, b_y)$, a *D-function* para este ponto e aresta é dada por:

$$D_{ABP} = (a_x - b_x)(a_y - p_y) - (a_y - b_y)(a_x - p_x).$$

Figura 1 – Restrições geométricas do problema.

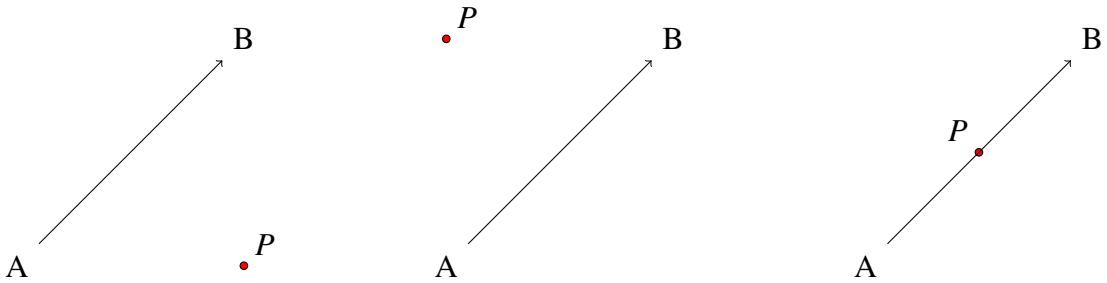


(a) Item fora do objeto. (b) Sobreposição entre itens.

Fonte: Elaborada pelo autor.

Assim, para uma aresta direcionada (orientada) da esquerda para a direita, $D_{ABP} < 0$ indica que o ponto P está à direita da aresta; $D_{ABP} > 0$ indica que o ponto P está à esquerda da aresta e $D_{ABP} = 0$ indica que o ponto P está sobre a aresta. Na Figura 2, essas três possibilidades são ilustradas.

Figura 2 – Exemplo de *D-function*.



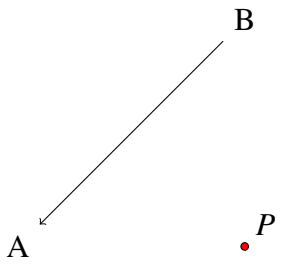
(a) $D_{ABP} < 0$. (b) $D_{ABP} > 0$. (c) $D_{ABP} = 0$.

Fonte: Elaborada pelo autor.

É importante destacar que caso a direção da aresta seja invertida ou a orientação do plano seja alterada, o valor de D_{ABP} também sofre alterações. Na Figura 3, note que o ponto está à esquerda da aresta $\overline{AB}$, porém segue que $D_{ABP} > 0$, pois a orientação da aresta está invertida em relação à Figura 2a.

Essa ferramenta é utilizada em modelos matemáticos, por exemplo, nos modelos de Scheithauer e Terno (1993), Cherri *et al.* (2016) e Rodrigues, Cherri e Mundim (2017).

Figura 3 – Exemplo de *D-function*.



$D_{ABP} > 0$

Fonte: Elaborada pelo autor.

2.1.2 No-fit polygon

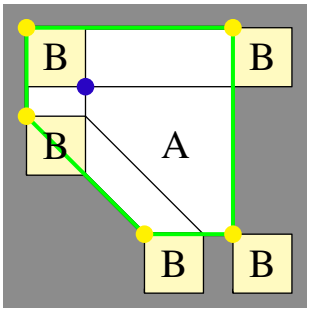
Outra ferramenta geométrica muito utilizada para verificar a sobreposição entre dois itens é o *no-fit polygon* (NFP). Dados dois itens i e j, o *no-fit polygon* entre eles (denotado por NFP_{ij}) é definido como o espaço tal que se i está fixado e o ponto de referência de j for posicionado em $p \in NFP_{ij}^{\circ}$ (interior de NFP_{ij}), então i e j se sobrepõem. Se o ponto de referência de j for posicionado em $p \in \partial NFP_{ij}$ (fronteira ou aresta de NFP_{ij}), então os itens se tocam. Se o ponto de referência de j for posicionado em $p \in \overline{NFP_{ij}}$ (complementar de NFP_{ij}), então os itens não se tocam.

Na Figura 4a, é ilustrado o *no-fit polygon* (polígono em branco) entre os itens A (fixo) e B (móvel), onde os pontos de referência dos itens estão destacados em azul (item A) e em amarelo (item B). As linhas em verde representam as arestas do NFP_{AB} e a região branca indica seu interior. Note que ao posicionar o item B em uma aresta de NFP_{AB}, ocorre contato com o item A (Figura 4b); quando o item B é posicionado fora de NFP_{AB} (região cinza), não há contato entre os dois itens. Entretanto, caso o item B seja posicionado no interior de NFP_{AB} (região branca), então ocorre a sobreposição entre os dois itens (Figura 4c).

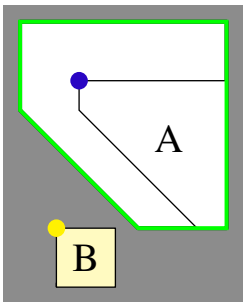
O *no-fit polygon* permite detectar a sobreposição entre os itens de forma rápida e precisa, pois tem-se que verificar apenas a posição do ponto de referência do primeiro item em relação a seu NFP com o segundo item. Calcular o *no-fit polygon* para itens convexos é simples, entretanto, ao considerar formas irregulares, o processo se torna significativamente mais difícil. Como Luo e Rao (2022) explica, existem diferentes algoritmos para se calcular o *no-fit polygon*. Por exemplo, o *sliding algorithm* desliza o item móvel em torno do item fixo para encontrar as arestas do *no-fit polygon*, enquanto algoritmos de decomposição buscam decompor os itens em partes convexas para gerar os *no-fit polygons* entre elas para posteriormente combiná-los no *no-fit polygon* entre os itens originais. Outra forma de se obter o *no-fit polygon* é por meio da soma de Minkowski das arestas.

Esta ferramenta geométrica é utilizada em modelos de otimização linear inteira-mista como em Toledo *et al.* (2013), Cherri *et al.* (2016), Leão *et al.* (2016), e também em heurísticas como em Pinheiro, Amaro-Junior e Saraiva (2016), Sato *et al.* (2019) e Sato *et al.* (2023).

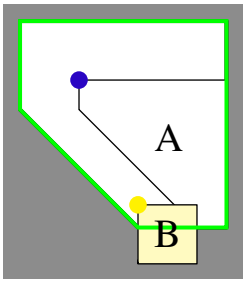
Figura 4 – *No-fit polygon* entre os itens A e B.



(a) Item B em cima da aresta de NFP_{AB}.



(b) Item B fora de NFP_{AB}.



(c) Item B dentro de NFP_{AB}.

Fonte: Elaborado pelo autor.

2.1.3 Inner-fit polygon

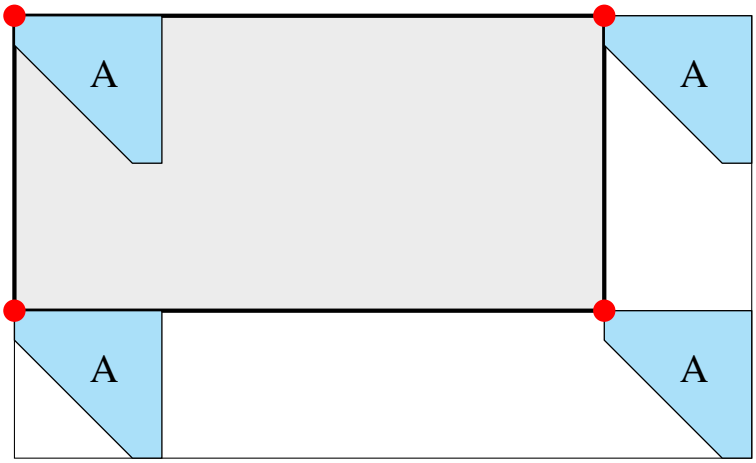
O *inner-fit polygon* é uma ferramenta geométrica utilizada para garantir que um item seja posicionado dentro do objeto de corte e não esteja em áreas defeituosas do objeto (regiões onde não pode haver itens). Assim, dado um objeto de corte e um item i, o *inner-fit polygon* de i (denotado por IFP_i) é definido como o espaço dentro do objeto tal que se o ponto de referência de i for posicionado, então o item estará totalmente contido no objeto e não ocupará uma área defeituosa. Na Figura 5, é ilustrado o *inner-fit polygon* de um item (considerando que o ponto de referência é o ponto em vermelho) em relação a um objeto de corte retangular.

Para se calcular o *inner-fit polygon*, é possível utilizar o *sliding algorithm* ou a soma de Minkowski de forma análoga ao *no-fit polygon* de modo que o item é móvel e o objeto de corte é o item fixo. Porém, no caso do *inner-fit polygon*, o *sliding algorithm* é utilizado para deslizar o item no interior do objeto de corte, enquanto que a soma de Minkowski é aplicada em relação ao complemento do objeto de corte. Vale destacar que no caso do objeto de corte ser um retângulo sem defeitos, o *inner-fit polygon* será também um retângulo.

2.1.4 Método raster

Esta ferramenta geométrica consiste em representar o objeto de corte e os itens por meio de matrizes. Assim, as entradas da matriz que representa o objeto cujos valores são diferentes de 0 representam uma região ocupada por um ou mais itens. Na Figura 6, é ilustrada a representação

Figura 5 – Exemplo de um *inner-fit polygon* para um dado item.

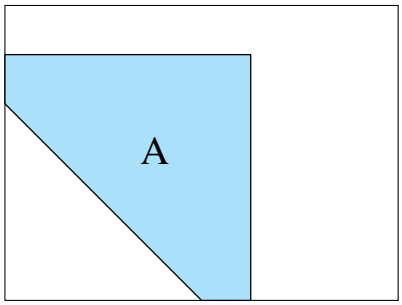


Fonte: Elaborada pelo autor.

binária do item A posicionado em um recipiente retangular. Desta forma, pode-se detectar sobreposição entre os itens ao somar os valores das entradas da matriz, pois quando o objeto de corte possui valores diferentes de 0 e 1, isso indica que alguns itens estão se sobrepondo.

A Figura 7 apresenta um exemplo de como o método *raster* é utilizado para verificar a ocorrência de sobreposição. Nas Figuras 7a e 7b, é ilustrado um plano de corte sem sobreposições e, portanto, não existem valores diferentes de 0 ou 1 na matriz. Já nas Figuras 7c e 7d, é apresentado um plano de corte com sobreposições, indicadas pelas entradas com valor igual a 2.

Figura 6 – Exemplo para uma representação por *raster method*.



(a) Item alocado dentro do objeto.

0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	1	0	0	0

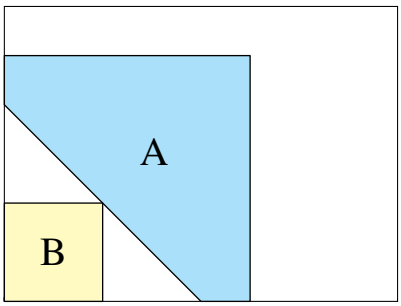
(b) Representação por *raster method*.

Fonte: Elaborado pelo autor.

A desvantagem do método *raster* é sua precisão que depende da densidade da matriz, pois itens irregulares podem necessitar de matrizes mais refinadas, o que acaba elevando o custo de memória do método ao considerar problemas de *nesting*. Por exemplo, na Figura 8, é ilustrada a representação de um mesmo item com malhas diferentes. Note que a representação da Figura 8b é mais precisa que a representação da Figura 8a, porém utiliza uma matriz mais densa para tal feito.

Devido à sua simplicidade de implementação, esta técnica é utilizada em heurísticas

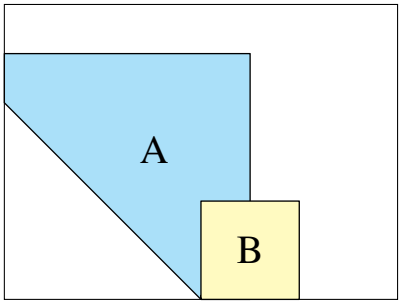
Figura 7 – Exemplos de planos de corte com e sem sobreposição representados utilizando o método *raster*.



(a) Plano de corte sem sobreposição

0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
1	1	0	1	1	0	0	0

(b) Representação pelo método *raster*.



(c) Plano de corte com sobreposição.

0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	2	1	0	0
0	0	0	1	2	1	0	0

(d) Representação pelo método *raster*.

Fonte: Elaborado pelo autor.

Figura 8 – Exemplo da representação com diferente matrizes.

1	1	1	0
1	1	1	0
0	1	1	0

(a) Malha mais grossa.

0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	1	0	0	0

(b) Malha mais densa

Fonte: Elaborado pelo autor.

como nos trabalhos de Mundim, Andretta e Queiroz (2017), Sato *et al.* (2019), Sato *et al.* (2023) e também no modelo de Baldacci *et al.* (2014).

2.2 Modelagem matemática

De acordo com Leão *et al.* (2020) existem três tipos de modelos lineares inteiro-mistos para o problema de *nesting*. O modelo com posicionamento contínuo permite que itens sejam alocados em qualquer lugar do recipiente (Scheithauer e Terno (1993), Dean (1996), Fischetti e Luzzi (2009), Alvarez-Valdés, Martinez e Tamarit (2013) e Cherri *et al.* (2016)). O modelo com

posicionamento discreto permite que os itens sejam posicionados apenas em lugares pré-definidos por uma malha de pontos (Toledo *et al.* (2013)). Enquanto o modelo com posicionamento semi-contínuo (Leão *et al.* (2016)) posiciona os itens em linhas, ou seja, enquanto uma das coordenadas (eixo x ou eixo y) é discretizada como no modelo de Toledo *et al.* (2013), a outra coordenada é mantida contínua como nos modelos de Scheithauer e Terno (1993), Dean (1996), Fischetti e Luzzi (2009), Alvarez-Valdés, Martinez e Tamarit (2013) e Cherri *et al.* (2016). Vale ressaltar que os modelos citados, com exceção de Scheithauer e Terno (1993), foram desenvolvidos para o problema de *Strip Packing*, ou seja, visam minimizar o comprimento utilizado de um recipiente com altura fixa. Scheithauer e Terno (1993) estudaram o problema de mochila bidimensional com itens irregulares, que visava maximizar a ocupação de um recipiente com dimensões fixas.

Além dos modelos citados, Leão *et al.* (2020) também apresentam modelos não-lineares e de programação por restrições, porém, estes não foram considerados nesta revisão bibliográfica. Uma das vantagens dos modelos não-lineares é considerar rotações contínuas, o que não é viável para o problema estudado, pois as partes das roupas quando rotacionadas definem diferentes orientações das fibras nos itens, o que nem sempre corresponde à orientação desejada para a fabricação do produto final. Além disso, caso a peça de roupa contenha listras (Alves (2016)) ou estampas, a rotação livre também se torna inviável. Outro trabalho desconsiderado foi o de Baldacci *et al.* (2014), pois ele estuda o problema de *nesting* considerando um recipiente de corte irregular.

2.2.1 Modelos contínuos

Dentre os modelos contínuos apresentados na literatura, como apontado por Leão *et al.* (2020), o trabalho de Cherri *et al.* (2016) reportava os melhores resultados até 2020. Os modelos contínuos permitem que os itens sejam posicionados em qualquer lugar do objeto, desde que não se sobreponham e estejam totalmente contidos dentro dele.

Em Cherri *et al.* (2016), são propostos dois modelos contínuos. O primeiro utiliza relações de trigonometria direta (*D-function*) para gerar restrições de não-sobreposição entre os itens, enquanto o segundo combina *D-functions* com *no-fit polygons* entre os itens para gerar as restrições de não-sobreposição. Detalhes sobre os conceitos de *D-function*, *no-fit polygon* podem ser encontrados em Bennell e Oliveira (2008).

O modelo baseado em *no-fit polygon* (denominado pelos autores como *NoFit Polygon Covering Model*) obteve os melhores resultados e, portanto, é apresentado neste subseção. Como mencionado, neste modelo, para verificar a não-sobreposição entre as peças, foi utilizado o conceito de *D-function* e de *no-fit polygon*.

Para definir as restrições de não-sobreposição, primeiro os autores decompõem os *no-fit polygons* entre os itens em partes convexas (quando necessário). Desta forma, $NFP_{ij} = \bigcup_{p=1,\ ...,Q_{ij}} NFP_{ij}^{p}$, em que NFP_{ij} é o *no-fit polygon* entre as peças i e j. Em seguida, para cada

parte $p = 1, \ldots, Q_{ij}$, são gerados os conjuntos K_{ij}^p que contêm as arestas orientadas da p-ésima parte do NFP_{ij}. Finalmente, a *D-function* é utilizada para verificar se o ponto de referência do item j (x_j, y_j) está fora do NFP_{ij} quando o item i está posicionado em (x_i, y_i). Para isso, são utilizadas as seguintes desigualdades:

$$D_{abp} = (a_{ij,x}^k - b_{ij,x}^k)(a_{ij,y}^k - g_{ij,y}) - (a_{ij,y}^k - b_{ij,y}^k)(a_{ij,x}^k - g_{ij,x}) \leq 0,$$

sendo $a_{ij,x}^k$, $b_{ij,x}^k$, $a_{ij,y}^k$ e $b_{ij,y}^k$ as coordenadas da aresta direcionada $k \in NFP_{ij}^p$, e $g_{ij,x} = x_i - x_j$ e $g_{ij,y} = y_i - y_j$ são as coordenadas em que o item i foi posicionado em relação ao item j (note que as coordenadas (x_i, y_i) foram transladadas pela posição (x_j, y_j) para que a *D-function* seja usada considerando as coordenadas fixas da aresta $\overline{ab}$). Essa desigualdade impõe que o item i esteja posicionado à direita ou acima da aresta $\overline{ab}$ de NFP_{ij}^p.

Considerando a constante:

$$\bar{C}_{pk}^{ij} = (a_{ij,x}^k - b_{ij,x}^k)a_{ij,y}^k - (a_{ij,y}^k - b_{ij,y}^k)a_{ij,y}^k \tag{2.1}$$

segue a seguinte desigualdade:

$$\bar{C}_{pk}^{ij} - (a_{ij,x}^k - b_{ij,x}^k)(y_i - y_j) + (a_{ij,y}^k - b_{ij,y}^k)(y_i - y_j) \leq 0.$$

Entretanto, apenas uma restrição deste tipo precisa estar ativa para cada NFP, logo a não-sobreposição entre os itens é dada por:

$$\bar{C}_{pk}^{ij} - (a_{ij,x}^k - b_{ij,x}^k)(y_i - y_j) + (a_{ij,y}^k - b_{ij,y}^k)(x_i - x_j) \leq (1 - v_{ij}^{pk})M_{ij}^{pk},$$
$$i, j \in \mathcal{N}, j > i, p \in Q_{ij},\ k \in K_{ij}^p.$$

$$\sum_{k \in K_{ij}^p} v_{ij}^{pk} = 1, \quad i, j \in \mathcal{N}, j > i,\ p \in Q_{ij}.$$

A variável binária v_{ij}^{pk} é igual a 1 se o item j está à direita ou acima da aresta $k \in NPF_{ij}^p$ e 0 caso contrário. M_{ij}^{pk} é uma constante suficientemente grande de forma que, caso $v_{ij}^{pk} = 0$, as desigualdade sejam sempre válidas.

O modelo de Cherri *et al.* (2016) é dado por (2.2)–(2.9). Antes, no entanto, são definidos seus conjuntos, parâmetros e variáveis.

Conjuntos

- $\mathcal{N}$ é o conjunto de itens, ou seja, $\mathcal{N} = \{1, \ldots, n\}$ em que n é último item;

- Q_{ij} é o conjunto de partes convexas em que o *no-fit polygon* entre os itens i e j é decomposto;
- K_{ij}^{p} é o conjunto de arestas direcionadas da p-ésima parte do *no-fit polygon* entre os itens i e j.

Parâmetros

- l_i^{min} e l_i^{max} são as distâncias entre a coordenada x do ponto de referência e os limites mínimo e máximo horizontais do item i, respectivamente;
- h_i^{min} e h_i^{max} são as distâncias entre a coordenada y do ponto de referência e os limites mínimo e máximo verticais do item i, respectivamente;
- M_{ij}^{pk} é um número suficientemente grande;
- $\bar{C}_{pk}^{ij}$ é uma constante do modelo definida em (2.1);
- $a_{ij,x}^{k}$, $b_{ij,x}^{k}$, $a_{ij,y}^{k}$ e $b_{ij,y}^{k}$ são parâmetros calculados com base nos *no-fit polygon* entre os itens i e j conforme definido anteriormente.

Variáveis

- $(x_i, y_i) \in \mathbb{R}^2$ corresponde a posição em que o item i é alocado;
- v_{ij}^{pk} é uma variável binária usada para impor que apenas uma restrição de não-sobreposição do tipo (2.5) seja satisfeita;
- z é uma variável contínua que representa o comprimento utilizado do objeto de corte para empacotar todos os itens.

$$\min \quad z \tag{2.2}$$

$$\text{s.a} \quad l_i^{min} \leq x_i \leq z - l_i^{max}, \qquad i \in \mathscr{N}, \tag{2.3}$$

$$h_i^{min} \leq y_i \leq H - h_i^{max}, \qquad i \in \mathscr{N}, \tag{2.4}$$

$$\begin{aligned} &\bar{C}_{pk}^{ij} - (a_{ij,x}^{k} - b_{ij,x}^{k})(y_i - y_j) + \\ &(a_{ij,y}^{k} - b_{ij,y}^{k})(x_i - x_j) \leq (1 - v_{ij}^{pk}) M_{ij}^{pk}, \end{aligned} \quad i, j \in \mathscr{N}, j > i, p \in Q_{ij},\ k \in K_{ij}^{p}, \tag{2.5}$$

$$\sum_{k \in K_{ij}^{p}} v_{ij}^{pk} = 1, \qquad i, j \in \mathscr{N}, j > i,\ p \in Q_{ij}, \tag{2.6}$$

$$z \geq 0, \tag{2.7}$$

$$(x_i, y_i) \in \mathbb{R}^2, \qquad i \in \mathscr{N}, \tag{2.8}$$

$$v_{ij}^{pk} \in \{0, 1\}, \qquad i, j \in \mathscr{N},\ p \in Q_{ij},\ k \in K_{ij}^{p}. \tag{2.9}$$

A função objetivo (2.2) minimiza o comprimento do objeto necessário para empacotar todos itens. Os conjuntos de restrições (2.3) e (2.4) garantem que os itens estejam dentro do recipiente. As restrições (2.5) utilizam trigonometria direta e os *no-fit polygons entre os itens* para impedir a sobreposição entre os itens. As restrições (2.6) impõem que apenas uma restrição (2.5) seja satisfeita para cada parte $p \in Q_{ij}$. O domínio das variáveis é dado por (2.7) – (2.9).

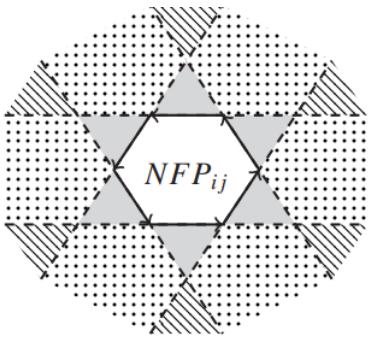
Este modelo pode gerar soluções de melhor qualidade quando comparado com os modelos de Toledo *et al.* (2013) e Leão *et al.* (2016), pois considera um espaço contínuo de busca. Entretanto, como cada item é tratado individualmente, instâncias com muitos itens (sejam eles do mesmo tipo ou não) resultam em muitas variáveis, o que pode levar a um elevado custo computacional para sua resolução.

2.2.1.1 Desigualdades válidas

Um problema do modelo proposto por Cherri *et al.* (2016) é a dificuldade para lidar com soluções simétricas. Por conta disso, Rodrigues, Cherri e Mundim (2017) propuseram novas restrições para eliminar essas soluções.

Rodrigues, Cherri e Mundim (2017) trataram 4 aspectos de simetria do modelo contínuo de Cherri *et al.* (2016), sendo eles a simetria causada pela variável binária v_{ij}^{pk}, por itens que podem ser transladados no objeto de corte e por itens do mesmo tipo. O primeiro caso de simetria ocorre porque as regiões definidas pelas variáveis v_{ij}^{pk} se sobrepõem, o que gera múltiplas soluções simétricas. A Figura 9 ilustra as regiões definidas pelas variáveis: as áreas em cor sólida representam a parte em que o item j está à direita de apenas uma aresta do NFP_{ij}, as áreas em pontilhado indicam a parte onde o item fica à direita de duas arestas e a parte listrada define a região em que fica à direita de três arestas. Note que, nesta situação, se o item j está na área listrada, existem três variáveis v_{ij}^{pk} que podem assumir o valor 1 e gerar a mesma solução final.

Figura 9 – Regiões definidas por v_{ij}^{pk}.

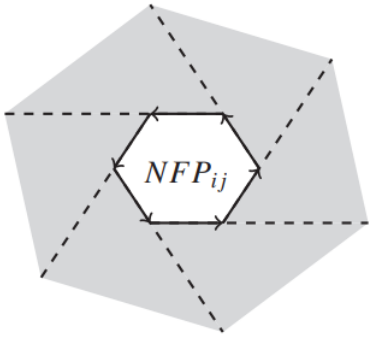


Fonte: Retirada de Rodrigues, Cherri e Mundim (2017).

Assim, Rodrigues, Cherri e Mundim (2017) propuseram adicionar as restrições (2.10). Estas restrições impõem que o item j também deve estar à direita da aresta k', em que k' é a aresta que precede a resta k. Desta forma, apenas uma variável v_{ij}^{pk} pode ser ativada. A Figura 10 ilustra as novas regiões.

$$\bar{C}^{ij}_{pk'} - (a^{k'}_{ij,x} - b^{k'}_{ij,x})(y_i - y_j) + (a^{k'}_{ij,y} - b^{k'}_{ij,y})(x_i - x_j) \leq (1 - v^{pk}_{ij})M^{pk'}_{ij}, \quad i,j \in \mathcal{N}, j > i, p \in Q_{ij},\ k \in K^p_{ij}. \tag{2.10}$$

Figura 10 – Novas regiões definidas por v^{pk}_{ij}.



Fonte: Retirada de Rodrigues, Cherri e Mundim (2017).

Outro fator que gera soluções simétricas é a possibilidade de transladar itens em uma solução. A translação é possível quando itens não se tocam ou quando nenhum item toca nas bordas do objeto de corte. Para impor que todos os itens estejam conectados, Rodrigues, Cherri e Mundim (2017) adicionaram a variáveis binárias ϕ_{ij} ao modelo. Esta variável é igual a 1 se o item i está conectado ao item j e 0 caso contrário. Portanto, foram adicionadas as restrições (2.11) que, em conjunto com as restrições (2.5) do modelo original, garantem que os itens i e j se toquem caso $\phi_{ij} = 1$.

$$\bar{C}^{ij}_{pk'} - (a^{k'}_{ij,x} - b^{k'}_{ij,x})(y_i - y_j) + (a^{k'}_{ij,y} - b^{k'}_{ij,y})(x_i - x_j) \geq -(1 - \phi_{ij})M^{pk'}_{ij}, \quad i,j \in \mathcal{N}, j > i, p \in Q_{ij},\ k \in K^p_{ij}. \tag{2.11}$$

Entretanto, apenas as restrições (2.11) não garantem que os itens estejam conectados. São necessárias as restrições (2.12) e (2.13), sendo (2.12) responsável por garantir que cada item está conectado pelo menos um outro e (2.13) responsável por impor que exista apenas um bloco de itens conectados. Entretanto, note que as restrições (2.13) utilizam o conjunto das partes dos itens $\mathscr{P}(S)$, o que causa um aumento exponencial na quantidade de restrições do modelo e, portanto, dificulta a sua resolução.

$$\sum_{j=1}^{i-1} \phi_{ji} + \sum_{j=i+1}^{|\mathcal{N}|} \phi_{ij} \geq 1, \quad i \in \mathcal{N}, \tag{2.12}$$

$$\sum_{i,j \in \mathscr{P}(S),\ i<j} \phi_{ij} \geq 1, \quad \forall S \subset \mathcal{N}, S \neq \emptyset. \tag{2.13}$$

A fim de tratar a simetria gerada pela possibilidade de transladar itens que não tocam o objeto de corte, Rodrigues, Cherri e Mundim (2017) impõem que pelo menos um item se

encontra na posição mais inferior do objeto. Para isso, são introduzidas as variáveis binárias τ_i, que valem 1 caso o item i esteja na posição mais inferior do objeto e 0 caso contrário. Assim, as restrições (2.14) e (2.15) garantem que exatamente um item se encontra na posição mais inferior.

$$y_i \leq h_i^{\min} + (H - h_i^{\max} - h_i^{\min})(1 - \tau_i), \quad i \in \mathscr{N}, \tag{2.14}$$

$$\sum_{i \in \mathscr{N}} \tau_i = 1. \tag{2.15}$$

Finalmente, o último fator que causa simetria é a existência de itens do mesmo tipo que podem ser permutados. Para eliminar este tipo de simetria, foram adicionadas as restrições (2.16) e (2.17), em que T_i representa o tipo do item i. As restrições (2.16) impede que itens do mesmo tipo sejam permutados, enquanto as restrições (2.17) garantem que apenas o item de menor índice de cada tipo pode ativar alguma das restrições (2.14).

$$y_i \leq y_j, \quad i \in \mathscr{N},\ j = \min\{k \in \mathscr{N}, k > i | T_i = T_k\}, \tag{2.16}$$

$$\tau_j = 0 \quad i \in \mathscr{N},\ j = \min\{k \in \mathscr{N}, k > i | T_i = T_k\}. \tag{2.17}$$

Os resultados mostraram que as novas restrições aumentam consideravelmente a eficiência no modelo, pois ao comparar com os resultados de Cherri *et al.* (2016), foi verificado que a nova versão do modelo provou a otimalidade para 19 instâncias, enquanto a versão antiga provou para 15.

2.2.2 Modelo discreto

Toledo *et al.* (2013) desenvolveram um modelo inteiro para o problema de *strip packing* baseado na representação do objeto por meio de uma malha quadriculada. Numa etapa de pré-processamento, o conjunto de pontos factíveis para o empacotamento de cada tipo de item é reduzido utilizando o conceito de *inner-fit polygon*, enquanto a sobreposição dos itens é analisada com o uso de *no-fit polygon*. A seguir é apresentado o modelo de Toledo *et al.* (2013).

Conjuntos

- $\mathscr{I}$ é o conjunto de tipos de itens, em que $i \in \mathscr{I}$ é o item do tipo i no conjunto;
- $\mathscr{B}$ é o conjunto de todos os pontos da malha;
- $\mathscr{B}_i$ é o subconjunto de pontos da malha para os quais a alocação de um item do tipo i é factível, ou seja, o item fica inteiramente contido no objeto. Em resumo, $\mathscr{B}_i$ corresponde aos pontos de $\mathscr{B}$ que pertencem ao *inner-fit polygon* do item do tipo i;
- NFP_{ij}^k é o subconjunto de pontos de $\mathscr{B}$ que compõem o *no-fit polygon* entre os itens de tipo i e j caso o item i seja alocada no ponto k.

Parâmetros

- d_i é a demanda do item do tipo i;
- c_i é a diferença entre a coordenada x do ponto de referência do item i e a coordenada x do vértice mais a direita deste mesmo item;
- x_k é a coordenada x do ponto k.

Variáveis

- δ_{ik} é igual a 1 se um item do tipo i é alocado no ponto k, 0 caso contrário;
- z é o comprimento total utilizado.

$$\min \quad z \tag{2.18}$$

$$\text{s. a} \quad \sum_{k \in \mathscr{B}_i} \delta_{ik} = d_i, \qquad i \in \mathscr{I}, \tag{2.19}$$

$$\delta_{jt} + \delta_{ik} \leq 1, \qquad t \in NFP^k_{ij}\ i, j \in \mathscr{I},\ j \geq i,\ k \in \mathscr{B}_i, \tag{2.20}$$

$$(x_k + c_i)\ \delta_{ik} \leq z, \qquad i \in \mathscr{I},\ k \in \mathscr{B}_i, \tag{2.21}$$

$$z \geq 0, \tag{2.22}$$

$$\delta_{ij} \in \{0,1\}, \qquad i \in \mathscr{I},\ j \in \mathscr{B}_i. \tag{2.23}$$

O modelo busca uma solução em que todos os itens sejam empacotadas no menor comprimento possível do objeto (2.18), ou seja, minimizando o comprimento do objeto definido pela variável z. As restrições (2.19) estabelecem que a demanda de cada item do tipo i seja atendida. A factibilidade do empacotamento dos itens no objeto é representada pelo conjunto de restrições (2.20), em que dois itens não podem se sobrepor e devem ser alocados de forma a estarem completamente no objeto ($k \in \mathscr{B}_i$). O comprimento utilizado do objeto é definido pela relação entre os pontos de alocação dos itens e a variável z. Esta relação é estabelecida pelo conjunto de restrições (2.21). O domínio das variáveis é definido em (2.22) e (2.23).

O desempenho desse modelo é afetada pela quantidade de pontos na discretização do objeto e pela quantidade de tipos de itens considerados. Assim, o modelo obtém bons resultados para instâncias com poucos tipos de itens e bastante repetições destes. A desvantagem do modelo é a necessidade de encontrar uma malha de pontos adequada, pois caso haja poucos pontos, a qualidade da solução pode ser altamente prejudicada, enquanto que muitos pontos elevam drasticamente o custo computacional do modelo.

2.2.3 Modelo de semi-contínuo

O modelo de Leão *et al.* (2016) considera linhas horizontais como os locais de posicionamento. Portanto, a coordenada x pode ser tratada como uma variável contínua, enquanto a coordenada y é discretizada. Logo, uma variável binária é utilizada para indicar em qual faixa o item está alocado e uma variável contínua define seu posicionamento na faixa. Desta forma, ele tem como proposta ser mais preciso que o modelo de Toledo *et al.* (2013), ao mesmo tempo que possui um custo computacional menor que o modelo de Cherri *et al.* (2016).

Parâmetros

- $\mathcal{N}$ é o conjunto de itens, ou seja, $\mathcal{N} = \{1...n\}$ em que n é o número de itens;
- P é a quantidade de faixas horizontais no modelo;
- r_i^{min} e r_i^{max} são, respectivamente, as coordenadas x mínima e máxima em que o item i pode ser alocado, ou seja, os limites da coordenada x do *inner-fit polygon* do item i;
- t_i^{min} e t_i^{max} são, respectivamente, as faixas mínima e máxima em que o item i pode ser alocado;
- M é uma constante suficientemente grande;
- n_{ij}^{min} e n_{ij}^{max} são as faixas mínima e máxima que o NFP_{ij} ocupa, ou seja, são os limites do intervalo da altura do NFP_{ij} quando este é posicionado na origem do objeto de corte;
- $C_{ij}^{s_j - s_i}$ é o número de concavidades que o NFP_{ij} possui entre as faixas s_i e s_j;
- a_{ij}^{kc} é a coordenada x do ponto mais a esquerda do NFP_{ij} na faixa k relacionado à concavidade c;
- b_{ij}^{kc} é a coordenada x do ponto mais a direita do NFP_{ij} na faixa k relacionado à concavidade c.

Variáveis

- $x_i \geq 0$ é uma variável contínua que representa a coordenada x em que o item i é alocado;
- $\delta_i^{s_i}$ é uma variável binária cujo valor é 1 quando o item i está posicionado na faixa s_i e 0 caso contrário;
- γ_{ij}^{c} é uma variável binária cujo valor é 1 se o ponto de referência do item i é posicionado à direita do item j enquanto está dentro da concavidade $c \in C_{ij}^{s_j - s_i}$ e 0 caso contrário.

$$\min \quad z \tag{2.24}$$

$$\text{s. a} \quad x_i + r_i^{max} \leq z, \qquad i \in \mathscr{N}, \tag{2.25}$$

$$x_i \leq x_j - b_{ij}^{(s_j - s_i)c} + M\,\gamma_{ij}^{c} + M\,(1-\delta_i^{s_i}) + M\,(1-\delta_j^{s_j}) \qquad i,j \in \mathscr{N}, \tag{2.26}$$
$$j > i,\ c = 1,\ ...,\ C_{ij}^{s_j - s_i},\ s_i = -t_i^{min},\ ...,\ P - t_i^{max}$$
$$s_j = -t_j^{min},\ ...,\ P - t_j^{max};\ n_{ij}^{min} \leq s_j - s_i \leq n_{ij}^{max},$$

$$x_i \geq x_j - a_{ij}^{(s_j - s_i)c} + M\,(1-\gamma_{ij}^{c}) + M\,(1-\delta_i^{s_i}) + M\,(1-\delta_j^{s_j}), \quad i \in \mathscr{N}, \tag{2.27}$$
$$j \in \mathscr{N},\ j > i;\ c = 1,\ ...,\ C_{ij}^{s_j - s_i};\ s_i = -t_i^{min},\ ...,\ P - t_i^{max},$$
$$s_j = -t_j^{min},\ ...,\ P - t_j^{max};\ n_{ij}^{min} \leq s_j - s_i \leq n_{ij}^{max},$$

$$\sum_{s=-t_i^{min}}^{P-t_i^{max}} \delta_i^{s} = 1, \qquad i \in \mathscr{N}, \tag{2.28}$$

$$-r_i^{min} \leq x_i, \qquad i \in \mathscr{N}, \tag{2.29}$$

$$\delta_i^{s_i} \in \{0,1\}, \qquad i \in \mathscr{N}, \tag{2.30}$$
$$s_i = -t_i^{min},\ ...,\ P - t_i^{max},$$

$$\gamma_{ij}^{c} \in \{0,1\}, \qquad i,\ j \in \mathscr{N}, \tag{2.31}$$
$$c = 1,\ ...,\ C_{ij}^{s_j - s_i}.$$

A função objetivo (2.24) busca minimizar o comprimento do objeto necessário para empacotar todos os itens. As restrições (2.25) associam o comprimento do objeto de corte com os itens alocados. As restrições (2.26) e (2.27) impedem que ocorra sobreposição: note que, se o item i é posicionado em x_i e na faixa s_i, e o item j é alocado entre as faixas que são atingidas pelo NFP_{ij} (ou seja, $n_{ij}^{min} \leq s_j - s_i \leq n_{ij}^{max}$), então a coordenada x do item j deve respeitar o limite da direita do NFP_{ij} (2.26) ou respeitar o limite da esquerda (2.27) para que não ocorra sobreposição. As restrições (2.28) impõem que todos os itens sejam empacotados e (2.29) garantem que os itens estejam totalmente contidos dentro do objeto de corte. O domínio das variáveis é definido por (2.30) e (2.31).

Os autores compararam os resultados obtidos com o modelo de Toledo *et al.* (2013). Para a maioria das instâncias testadas, o modelo obteve resultados de qualidade melhor ou igual que os obtidos por Toledo *et al.* (2013). Porém, também foi observado um aumento no custo computacional, o que ocasionou no tempo limite de resolução sendo atingido em várias instâncias.

2.3 Métodos heurísticos

Como destacam Bennell e Oliveira (2009), o problema de empacotamento de peças irregulares tem alta complexidade computacional e, portanto, muitos métodos heurísticos são propostos para sua solução. Os autores apresentam um levantamento de heurísticas utilizadas para tratar o problema até o ano de 2009. Após este ano, os melhores resultados da literatura foram apresentados por Elkeran (2013). Então, a revisão apresentada nesta seção está focada em trabalhos de 2014 até 2023, incluindo o trabalho de Elkeran (2013).

Elkeran (2013) propõe uma heurística de duas fases denominada *guided cuckoo search* (GCS) para o problema de *strip packing*, que tem como objetivo empacotar todos os itens demandados em um objeto de altura fixa enquanto minimiza o comprimento necessário. Além disso, Elkeran (2013) permitiu rotações discretas para os itens. A heurística GCS faz uso de operações de compressão e eliminação de sobreposição para realizar melhorias na solução em cada iteração. A operação de compressão é feita da seguinte forma: dada uma solução inicial factível, é diminuído o comprimento do objeto de corte e os itens são realocados de modo a minimizar a sobreposição entre eles. Caso a solução obtida não possua sobreposição, então ela é armazenada e novamente é feita uma tentativa de compressão. Caso ela apresente sobreposição, é realizada uma tentativa de factibilização desta solução: para isso, o autor propõe uma heurística combinando as meta-heurísticas *cuckoo search* e *guided local search* para realocar os itens novamente. Vale ressaltar que a operação de eliminação de sobreposição pode não ser bem sucedida e, nesse caso, é permitido um aumento no comprimento do objeto para em seguida aplicar os procedimentos da etapa de compressão. Para iniciar o processo de melhoria, é necessária uma solução inicial. O autor utiliza a heurística construtiva *bottom-left* combinada com a ferramenta *pairwise clustering* que consiste em agrupar itens que se encaixem bem de forma que sejam tratados como um só, pois isso reduz o custo computacional do problema.

Diferentes autores tratam o problema utilizando algoritmos genéticos, como Pinheiro, Amaro-Junior e Saraiva (2016), Mundim, Andretta e Queiroz (2017), Souza Queiroz e Andretta (2020) e Lu, Hu e Ng (2023).O método proposto por Pinheiro, Amaro-Junior e Saraiva (2016) utiliza a meta-heurística *random-key genetic algorithm* (RKGA) para resolver o problema de *strip packing*. Cada cromossomo possui tamanho $3n$ (em que n é a quantidade total de itens empacotados). Os n primeiros alelos determinam a sequência de alocação dos itens, os n alelos seguintes determinam a rotação discreta que o item se encontra e os alelos restantes correspondem à regra de posicionamento utilizada (*bottom-left, greedy-bottom-left* e *no-fit polygon-based heuristic*). Os resultados obtidos foram comparados com os de Elkeran (2013) e se mostraram inferiores, porém, como os autores ressaltam, esta heurística permite o uso de diversas regras de posicionamento, sendo assim possível utilizar outras heurísticas construtivas para melhorar sua performance.

Em Mundim, Andretta e Queiroz (2017), é utilizada a meta-heurística *biased random-key genetic algorithm* (BRKGA) para resolver o problema de *strip packing* e também com duas

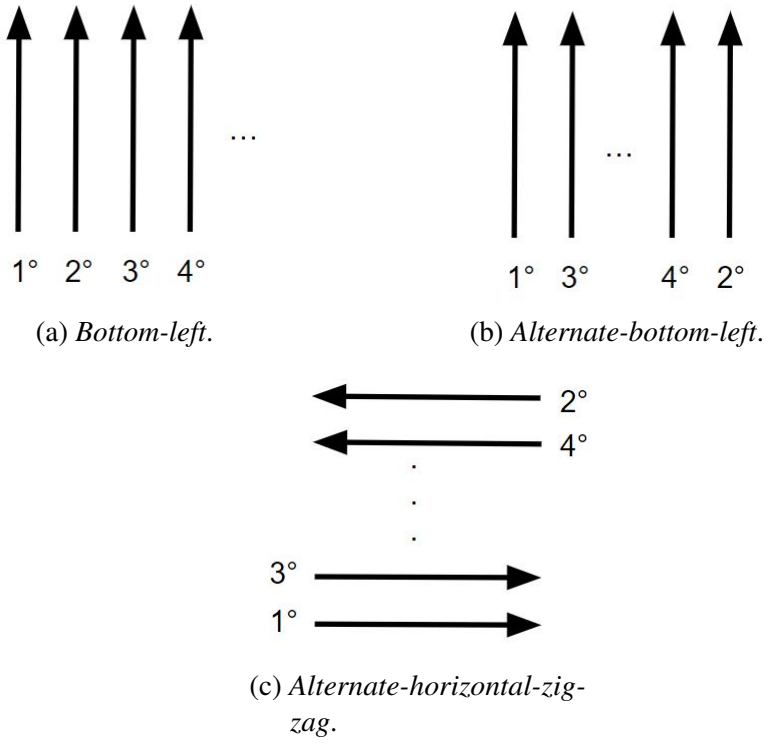
dimensões abertas (altura e comprimento variáveis). Diferentemente do que foi feito em Pinheiro, Amaro-Junior e Saraiva (2016), o cromossomo neste caso possui tamanho n e controla a ordem de posicionamento dos itens e a rotação discreta deles. A regra de posicionamento é sempre a heurística *bottom-left*. Os resultados obtidos foram comparados com Elkeran (2013) e Pinheiro, Amaro-Junior e Saraiva (2016) e foi constatado que a heurística é superior à proposta por Pinheiro, Amaro-Junior e Saraiva (2016), porém não a de Elkeran (2013). Entretanto, é válido destacar que esta foi a primeira heurística proposta para o problema de *strip packing* com duas dimensões abertas, sendo capaz de resolver instâncias médias e grandes da literatura.

Souza Queiroz e Andretta (2020) propuseram um BRKGA para o problema de mochila bidimensional com itens irregulares (*two-dimensional irregular knapsack*) e, portanto, não é possível comparar os resultados com os trabalhos citados anteriormente. Desta forma, as autoras comparam os resultados com o trabalho de Mundim *et al.* (2018), que era considerado o estado da arte quando o trabalho foi desenvolvido. Nesta heurística, o cromossomo possui tamanho $(n+p+1)$, sendo n alelos responsáveis por definir a ordem de posicionamento e rotação discreta dos itens, p alelos para indicar a quantidade de posições ignoradas na regra de posicionamento para cada item e 1 alelo responsável por definir a probabilidade do cromossomo filho herdar alelos do pai elite. Em relação ao posicionamento dos itens, foram utilizadas as heurísticas *bottom-left* (Figura 11a), a *alternate-bottom-left* e a *alternate-horizontal-zig-zag*. Na *alternate-bottom-left*, em vez de priorizar as posições inferiores mais à esquerda, são alternadas posições inferiores à esquerda com inferiores à direita como ilustra a Figura 11b. Já a *alternate-horizontal-zig-zag*, a ordem de posicionamento corresponde a um *zigzag*, como é exemplificado pela Figura 11c. Assim, combinando o BRKGA com cada regra de posicionamento, foi obtida uma versão da heurística. Com base nos testes, concluiu-se que a regra *alternate-horizontal-zig-zag* é a melhor, seguida da regra *bottom-left*.

Vale destacar que além do BRKGA, Souza Queiroz e Andretta (2020) também propuseram o uso da meta-heurística *Variable Neighborhood Search* (VNS). Esta heurística representa as soluções de forma semelhante ao BRKGA (exceto que não possui o alelo responsável pela probabilidade de herança) e utiliza como vizinhanças soluções que contenham permutação entre os alelos (ou seja, os valores dos alelos i e j são permutados) e inserção de um alelo logo após outro (ou seja, reposicionar o alelo i logo após o alelo j). Em relação à regra de posicionamento, o uso da *alternate-horizontal-zig-zag* se mostrou mais eficiente, seguida da regra *bottom-left*. Os resultados tanto do BRKGA quanto do VNS se mostraram bastante competitivos em relação à literatura, sendo superiores ou tão bom quanto todos os resultados obtidos por Mundim *et al.* (2018). Porém, ao comparar o desempenho do BRKGA com o VNS, foi observada uma melhor performance da heurística BRKGA por ser computacionalmente mais eficiente.

Lu, Hu e Ng (2023) estudam o problema de impressão 3D. Mais especificamente, o objetivo dos autores é empacotar um conjunto de formas tridimensionais utilizando o menor número de máquinas de impressão. Este problema pode ser simplificado ao projetar estas formas

Figura 11 – Regras de posicionamento de Souza Queiroz e Andretta (2020).



(a) *Bottom-left*.

(b) *Alternate-bottom-left*.

(c) *Alternate-horizontal-zig-zag*.

Fonte: Elaborado pelo autor.

no plano para obter figuras bidimensionais. Assim, para resolver o problema, é necessário solucionar o problema de *strip packing* rapidamente como sub-rotina. Portanto, os autores propõem uma heurística denominada *pixel-based additive manufacturing packing algorithm* (PAMPA), que usa a heurística *bottom-left* para posicionar os itens e o BRKGA para otimizar a ordem de posicionamento destes. Vale ressaltar que o *bottom-left* implementado pelos autores (denominado de *accelerated bottom-left*) considera rotações discretas para os itens e calcula o melhor ângulo de posicionamento para cada item (ou seja, de forma que sejam obtidos encaixes mais eficiente). Diferente dos trabalhos anteriores, Lu, Hu e Ng (2023) priorizam a velocidade de obtenção das soluções, então o critério de parada da heurística é a obtenção de uma solução cujo comprimento é menor ou igual a um dado limite. Os resultados obtidos para algumas instâncias foram comparados com os de Sato *et al.* (2019) (trabalho este que possui os resultados mais próximos de Elkeran (2013)), onde o PAMPA obteve resultados ligeiramente inferiores, porém com um tempo de execução muito menor (a instância mais demorada levou 2,8 segundos), mostrando assim um potencial competitivo nesta heurística.

Na Tabela 1, são resumidas as informações dos algoritmos genéticos citados. A coluna "Artigo" identifica o trabalho em questão, "Heurística" indica a heurística implementada, "Tam. crom." se refere ao tamanho do cromossomo, "Características Cromossomo" indica as características do problema que o cromossomo representa e "RdP" indica a regra de posicionamento (ou heurística construtiva) utilizada.

Tabela 1 – Resumo dos algoritmos genéticos.

Artigo	Heurística	Tam. crom.	Características Cromossomo	RdP
Pinheiro, Amaro-Junior e Saraiva (2016)	RKGA	$3n$	Sequência de inserção dos itens, rotação dos itens e regra de posicionamento.	*Bottom-left, greedy-bottom-left* ou *no-fit polygon-based heuristic*.
Mundim, Andretta e Queiroz (2017)	BRKGA	n	Sequência de inserção dos itens, rotação dos itens.	*Bottom-left*.
Souza Queiroz e Andretta (2020)	BRKGA	$n+p+1$	Sequência de inserção dos itens, rotação dos itens, posições de alocação ignoradas e determina a probabilidade de herança.	*Alternate-horizontal-zig-zag* (melhores resultados), *bottom-left* (resultados intermediários) e *alternate-bottom-left* (piores resultados).
Lu, Hu e Ng (2023)	BRKGA	n	Sequência de inserção dos itens.	*Accelerated bottom-left*.

Além dos métodos baseados em algoritmos genéticos, tem-se o trabalho de MirHassani e Bashirzadeh (2015) que propuseram uma meta-heurística *greedy randomized adaptive search procedures* (GRASP) e sua versão reativa (RGRASP) (versão esta que adapta os parâmetros da heurística ao longo das iterações) para resolver o problema de *strip packing*. Vale ressaltar que rotações não foram permitidas. A heurística utiliza os conceitos do modelo matemático de Toledo *et al.* (2013) para verificar a sobreposição entre as peças. De forma resumida, o modelo de Toledo *et al.* (2013) discretiza o objeto de corte com uma malha regular de pontos e permite o posicionamento dos itens apenas neles. As restrições de não-sobreposição são construídas de forma que se o item i é posicionado no ponto d, então são criadas restrições que impedem o posicionamento de qualquer outro item em pontos que causem sobreposição com o item i quando este se encontra no ponto d. Para representar a solução, MirHassani e Bashirzadeh (2015) codificam todos os pares *(tipo de item, ponto da malha)* com um número, pois desta forma é criado um vetor binário de tamanho igual à quantidade de pares, em que as entradas iguais a 1 indicam que o item de determinado tipo é posicionado no ponto em questão e 0 indica o contrário. Na etapa construtiva do GRASP, é priorizado o uso de pontos que pouco limitam o posicionamento dos demais itens. Para avaliar quais pontos são menos restritivos, os autores contaram a quantidade de vezes que cada ponto aparecia nas restrições do modelo, pois, desta forma, pontos que mais aparecem são mais restritivos. Em relação à busca local, é utilizado a vizinhança de *k-p exchange*, em que k entradas do vetor solução que valem 1 são mudadas para 0 e p entradas que valem 0 são mudadas para 1. Os autores consideraram apenas as trocas 0-1, 1-1, 1-2 e 2-2, pois valores maiores acarretam um custo computacional maior. Para evitar que as soluções sejam infactíveis, no final é feita uma troca 0-n para factibilizar a solução (ou seja, caso algum item esteja faltando na solução, ele é adicionado). Os resultados foram comparados com os obtidos por Toledo *et al.* (2013) e se mostraram competitivos, pois, embora exista uma queda na qualidade das soluções, o tempo de execução foi reduzido drasticamente em instâncias maiores.

Sato, Martins e Tsuzuki (2016) estudam o problema de *strip packing* com rotações discretas e propõem a heurística *pairwise exact placement algorithm* (PEPA). A heurística utiliza os conceitos de alocação exata e alocação móvel para posicionar os itens listados em sequência.

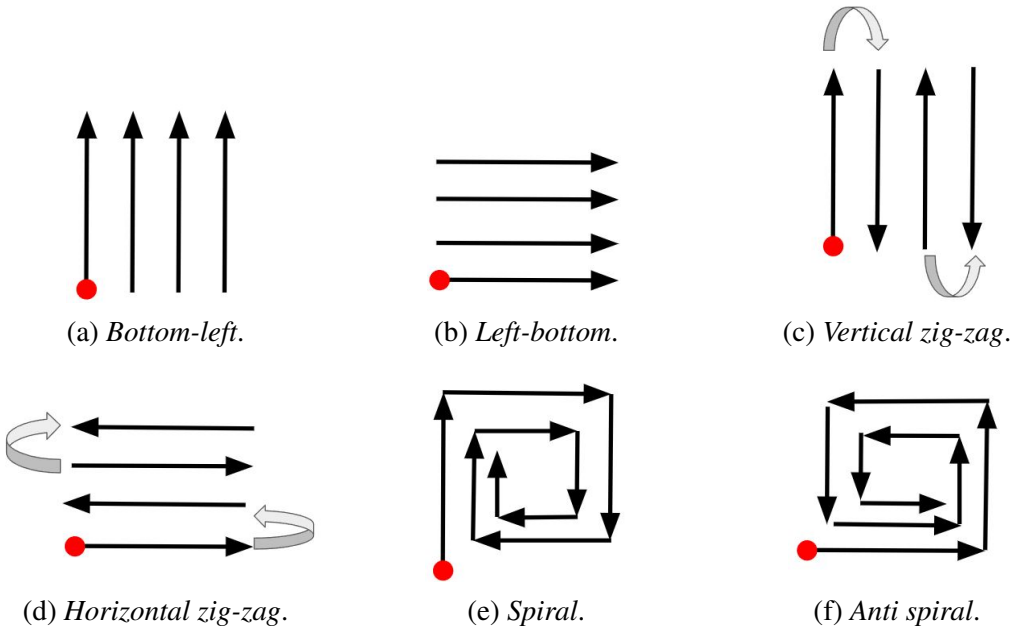
A alocação exata de um item ocorre quando este não pode ser movido de sua posição atual sem que se sobreponha com outro item no processo. Já a alocação móvel de um item ocorre quando o item pode ser movido de sua posição ao longo de uma direção sem que acorra sobreposição com outros itens. Os autores notaram que ambos os conceitos ajudam a criar planos de cortes mais eficientes. Entretanto, ao alocar um item por vez no objeto de corte, não é possível garantir que ele seja alocado de forma exata ou móvel. Sendo assim, em cada iteração de alocação, os itens são posicionados em pares, de forma a almejar o encaixe exato ou móvel do segundo item do par, sendo o encaixe exato priorizado sobre o móvel; caso não seja possível obter nenhum tipo de alocação, posiciona-se o primeiro item do par através da heurística *left-bottom*, enquanto o segundo item do par é alocado na próxima iteração formando par com o item seguinte. Os resultados foram comparados com Elkeran (2013), porém se mostraram inferiores. Entretanto, a heurística fornece uma aplicação dos conceitos de alocação exata e móvel que, como os autores afirmam, ainda são pouco estudados.

Mundim *et al.* (2018) propõem uma heurística genérica para resolver os problemas de *placement, cutting stock, knapsack* e *bin packing* com rotações discretas. A heurística é denominada de H4NP. Em cada iteração, são escolhidas, de forma aleatória, uma regra de posicionamento e uma sequência para os itens serem alocados. Além disso, em cada iteração, é atualizada a melhor solução. Em relação às regras de posicionamento, foram utilizadas seis: *bottom-left, left-bottom, vertical zig-zag, horizontal zig-zag, spiral* e *anti spiral*. A Figura 12 mostra a prioridade de cada regra de posicionamento: o ponto preto indica a posição de origem, enquanto as setas mostram a direção que a regra toma. Como os problemas não incluem o *strip packing*, os resultados não foram comparados com Elkeran (2013) e sim comparados com trabalhos que estudaram os problemas listados. Para os problemas de *placement, cutting stock* e *knapsack*, a heurística obteve resultados superiores aos da literatura, enquanto para o problema de *bin packing*, os resultados foram competitivos.

Chehrazad, Roose e Wauters (2022) desenvolveram uma nova versão da heurística *bottom-left* que utiliza uma semi-discretização: em vez de discretizar o objeto de corte e os itens com o *raster method*, são utilizadas linhas verticais para isso. Assim, é obtida uma maior precisão nas soluções. Os resultados mostraram que a heurística se mantém rápida como a *bottom-left* discreta e apresenta melhores resultados que ela. Embora a qualidade dos resultados não seja tão boa em comparação com os trabalhos anteriores, este algoritmo pode ser utilizado como uma heurística construtiva no lugar da *bottom-left* discreta.

Os trabalhos de Sato *et al.* (2019) e Sato *et al.* (2023) utilizam os conceitos de *obstruction map* e *raster penetration map* para realizar operações de compressão seguidas de eliminação de sobreposição. De maneira geral, o *obstruction map* discretiza o objeto de corte e os itens posicionados nele com uma matriz (método *raster*); em seguida, um item é escolhido e as posições dentro de seu *inner-fit polygon* são avaliadas de modo que indiquem o quanto de sobreposição a alocação do item naquela posição causa. No caso, a quantidade de sobreposição

Figura 12 – Regras de posicionamento de Mundim *et al.* (2018).



(a) *Bottom-left*. (b) *Left-bottom*. (c) *Vertical zig-zag*.

(d) *Horizontal zig-zag*. (e) *Spiral*. (f) *Anti spiral*.

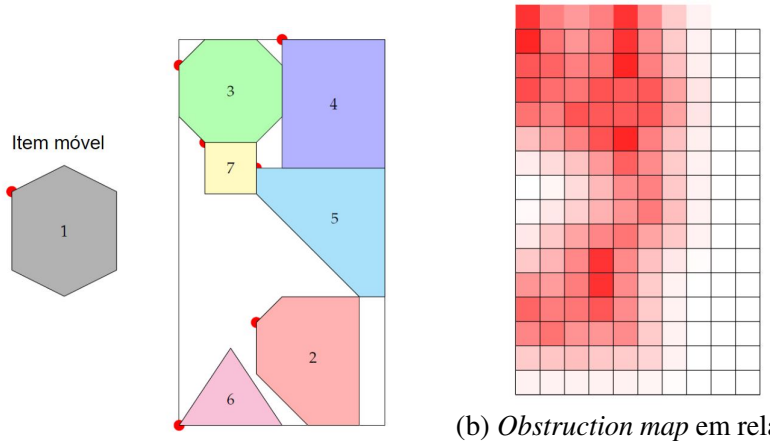
Fonte: Elaborado pelo autor.

é medida pela quantidade de movimentos necessários (translação do item) para eliminar a sobreposição. A Figura 13 ilustra o uso do *obstruction map* para posicionar um item dentro do objeto sem que ocorra sobreposição; note que quanto mais escuro é o ponto, maior é a sobreposição causada.

Já o *raster penetration map* é uma versão do *obstruction map*, porém considerando o *no-fit polygon* entre dois itens. Em vez de utilizar o objeto de corte com itens posicionados, é criado um *obstruction map* do *no-fit polygon* para avaliar o quanto os itens se sobrepõem caso violem o NFP. A Figura 14 ilustra um exemplo de como o *raster penetration map* é construído: note que na Figura 14b, quanto mais escuro, maior é a sobreposição causada ao alocar o item j naquela posição. Este conceito é utilizado para agilizar o calculo do *obstruction map*.

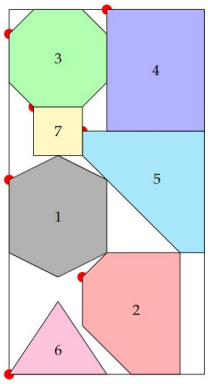
Baseado nestes conceitos, Sato *et al.* (2019) propõem uma heurística denominada *raster overlap minimization algorithm* (ROMA) para o problema de *strip packing* capaz de comprimir (melhorar) e eliminar sobreposição (factibilizar) de uma solução a cada iteração. Resumindo, a heurística diminui o comprimento do objeto de corte utilizando o *obstruction map* para realocar os itens visando minimizar a sobreposição; as novas posições são determinadas com base no *obstruction map*. Caso a solução obtida não seja factível, é permitido um aumento no comprimento do objeto até que a sobreposição seja totalmente eliminada. Este processo é repetido até que um critério de parada seja atingido. Vale destacar que foram permitidas rotações discretas dos itens. Os resultados se mostraram promissores quando foram comparados com os obtidos por Elkeran (2013) e se mostraram melhores ou iguais em 9 de 15 instâncias comparadas, sendo assim a heurística mais próxima do estado da arte.

Figura 13 – Exemplo de *obstruction map*.



(a) Solução inicial.

(b) *Obstruction map* em relação ao item móvel.

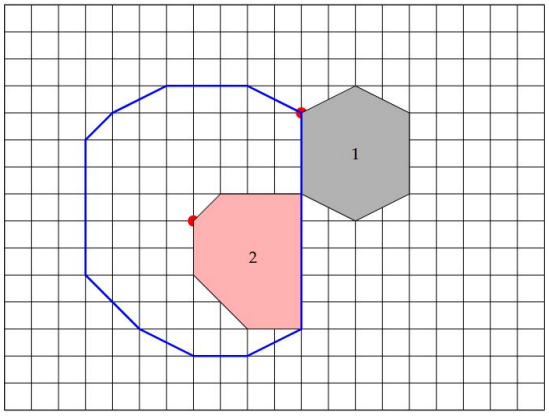


(c) Item posicionado sem sobreposição.

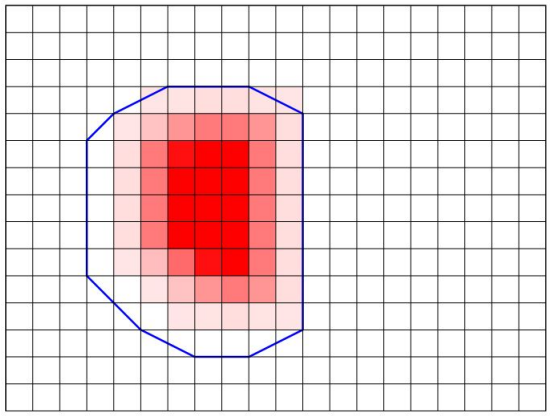
Fonte: Elaborado pelo autor.

Sato *et al.* (2023) propõem a heurística chamada de *fast obstruction map local search* (FOMLS), que utiliza o modelo de Toledo *et al.* (2013) para tratar a alocação dos itens no objeto de corte em vez do método *raster*. Para eliminar a sobreposição gerada pela etapa de compactação, os autores utilizam o *obstruction map* e o *raster penetration map* para realocar os itens de modo a minimizar a sobreposição. Caso não seja possível eliminar totalmente a sobreposição, é permitido um aumento no tamanho do objeto de corte para em seguida ser feita uma outra tentativa de factibilizar da solução. Diferentemente de Sato *et al.* (2019), Sato *et al.* (2023) estudam os problemas de *strip packing* com uma dimensão aberta, com duas dimensões abertas e com o objeto de corte quadrado (ou seja, o objeto possui altura e comprimento variáveis e de mesmo valor), todos eles considerando rotações discretas para os itens. Desta forma, os resultados do *strip packing* com uma dimensão foram comparados com Elkeran (2013), enquanto os resultados com duas dimensões foram comparados com Mundim, Andretta e Queiroz (2017). Até a data de publicação, não existiam trabalhos para realizar comparações em relação ao problema de *strip packing* quadrado, então os autores propuseram um modelo de programação linear inteira-mista para o problema e compararam seus resultados com a heurística FOMLS.

Figura 14 – Exemplo de *raster penetration map*.



(a) *No-fit polygon* discretizado com o *raster method*.



(b) *Raster penetration map*.

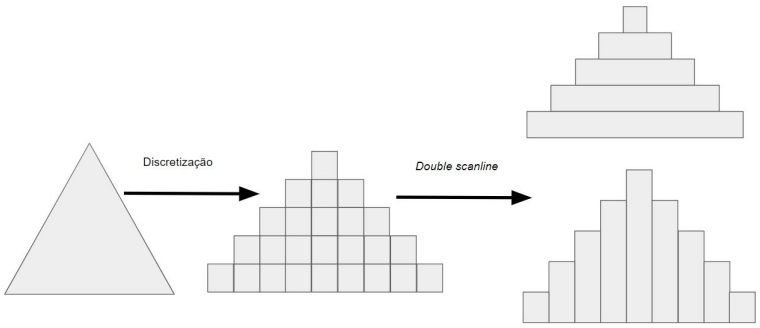
Fonte: Elaborado pelo autor.

Para resolver o modelo, os autores utilizaram o solver CPLEX 12.7. Em relação ao *strip packing* com uma dimensão, FOMLS obteve soluções de qualidade melhor ou igual em 8 de 15 instâncias ao ser comparado com Elkeran (2013), porém utilizando um tempo de execução maior. Para o problema de *strip packing* com duas dimensões, FOMLS obteve resultados melhores ou iguais em 14 de 15 instância, o que consagra estes resultados como os melhores da literatura atual. Os resultados para o problema de *strip packing* com objeto de corte quadrado mostraram que a heurística FOMLS consegue atingir o resultado ótimo obtido pelo solver, além de que encontra melhores soluções que o solver quando este não atinge a otimalidade.

Umetani e Murakami (2022) propõem a heurística denominada de *guided coordinate descent heuristics* (GCDH) para o problema de *strip packing* com rotações discretas. Esta heurística realiza operações de compressão e eliminação de sobreposição para melhorar uma solução a cada iteração, assim como Sato *et al.* (2019) e Sato *et al.* (2023). Assim, a GCDH faz uma tentativa a cada iteração de comprimir a solução diminuindo o comprimento do objeto de corte e espalhando os itens aleatoriamente dentro dele para, em seguida, realocar os itens de forma a minimizar a sobreposição. Caso a nova solução seja factível, outra tentativa de melhoria é

realizada. Caso não seja possível eliminar totalmente as sobreposições, é permitido um aumento no comprimento do objeto e os itens são novamente espalhados aleatoriamente dentro dele para ser realizada outra tentativa de factibilização. Para realizar o processo de eliminação de sobreposição, a heurística move os itens na solução verticalmente ou horizontalmente de modo que minimize a quantidade de sobreposição. Um ponto importante a se destacar é que os autores utilizam uma representação chamada de *double scanline*, que representa os itens utilizando um par de discretizações por barras, sendo uma discretização com barras verticais e outra com barras horizontais. Essa representação reduz o custo computacional gerado pelo método *raster* e torna mais fácil as operações de translação vertical e horizontal dos itens. A Figura 15 ilustra um exemplo de como converter a representação por *raster* para a *double scanline*. Os resultados foram comparados com os de Elkeran (2013) e Sato *et al.* (2019), porém se mostraram inferiores em ambos os casos. Entretanto, a representação por *double scanline* se mostrou eficiente para a heurística implementada, uma vez que diminuiu o custo de memória do problema e acelerou o cálculo das translações.

Figura 15 – Representação *double scanline*.



Fonte: Elaborado pelo autor.

Martinez-Sykora *et al.* (2017) propõem o uso de uma *math*-heurística para a resolução do *bin packing* com rotações discretas. Como os autores citam, existiam poucos trabalhos que tratavam o *bin packing* considerando itens irregulares e, portanto, o trabalho não apresenta comparações com outros resultados. Vale destacar que Mundim *et al.* (2018) também trabalhou com o *bin packing*, porém utilizaram instâncias diferentes. O método proposto pelos autores pode ser descrito em dois passos. No primeiro passo, o problema é reduzido a uma dimensão, ou seja, os itens e o objeto de corte são representados pelas suas respectivas áreas. Desta forma, é resolvido este problema simplificado para estimar quantos objetos de corte serão necessários para empacotar todos os itens e também para designar cada item a um objeto. No fim do primeiro passo, é obtida uma lista enumerada de objetos com seus respectivos itens. Já no segundo passo, tenta-se empacotar todos os itens designados ao primeiro objeto utilizando um modelo de programação linear inteira-mista. Caso a alocação de todos os itens seja possível, este plano de corte é considerado como completo e é removido da lista; assim, repete-se o segundo passo para o objeto seguinte da lista. Entretanto, caso não seja possível empacotar todos os itens, o

objeto de corte é preenchido de forma factível e este plano de corte é considerado completo; os itens que foram empacotados são descontados das demandas e é aplicado o primeiro passo nos itens restantes para obter uma nova estimativa dos planos de corte e depois é aplicado o segundo passo na nova lista de objetos.

Zhang *et al.* (2022) estudaram o problema de *bin packing* com rotações discretas. Os autores utilizam uma heurística construtiva para gerar uma solução inicial e então realizaram uma busca local. Para a fase construtiva, os autores propuseram a heurística *waste least first decreasing* (WLFD), que designa cada item a um *bin* de forma a obter *bins* com maior ocupação. Em cada iteração, um item é alocado de forma a maximizar a ocupação de um *bin*; para isso, tenta-se posicionar o item no *bin* de maior ocupação. Caso não seja possível, tenta-se posicioná-lo em outros *bins* seguindo a regra de maximizar a ocupação; se não for possível alocá-lo em nenhum, é criado um novo *bin* na solução. Como regra de posicionamento, Zhang *et al.* (2022) utilizaram o *bottom-left* e, caso não seja possível posicionar o item, são feitas iterações de compressão e eliminação de sobreposição para tentar a alocação. Após obter uma solução inicial, uma busca local é feita ao tentar trocar itens entre os *bins* de forma a obter objetos com maior ocupação. Ou seja, trocam-se itens entre os *bins* mais ocupados e os menos ocupados com o objetivo de melhorar ainda mais o uso dos *bins* com maior ocupação. Os resultados foram comparados com Martinez-Sykora *et al.* (2017) e se mostraram melhores na maioria das instâncias, sendo assim este é o trabalho estado da arte até o momento.

Na Tabela 2, é apresentado um resumo de todas as heurísticas citadas nesta seção. Os trabalhos que apresentaram os melhores resultados estão destacados em negrito.

2.4 Considerações

Em relação aos métodos heurísticos, poucos trabalhos estudam o *bin packing problem*, tema este que mais se aproxima do problema estudado neste trabalho. Os melhores resultados para este problema fora obtidos por Zhang *et al.* (2022). Entretanto, muitos trabalhos foram propostos para o *strip packing problem*, sendo Elkeran (2013) e Sato *et al.* (2019) os que apresentam os melhores resultados.

Considerando os modelos de programação matemática, para problemas com muitas repetições do mesmo item, o modelo de Toledo *et al.* (2013) é o mais adequado para este trabalho, pois o número de variáveis binárias deste modelo é proporcional à quantidade de tipos de itens. Por outro lado, o número de variáveis dos modelos de Cherri *et al.* (2016) e Leão *et al.* (2016) dependem do número total de itens e do número de arestas de seus *no-fit polygons*. Como, no contexto de confecções, tem-se muitas peças iguais, optou-se por adaptar o modelo de Toledo *et al.* (2013) para o problema estudado.

Vale destacar que, na literatura, existem trabalhos que propõem melhorias do modelo de Toledo *et al.* (2013). Segundo Cherri *et al.* (2018), o uso de malha regulares nem sempre é

Tabela 2 – Resumo das heurísticas citadas.

Artigo	Heurística	Problema	Rotação
Elkeran (2013)	GCS	*Strip packing*	Discreta
MirHassani e Bashirzadeh (2015)	GRASP	*Strip packing*	Não permitida
Sato, Martins e Tsuzuki (2016)	PEPA	*Strip packing*	Discreta
Pinheiro, Amaro-Junior e Saraiva (2016)	RKGA	*Strip packing*	Discreta
Mundim, Andretta e Queiroz (2017)	BRKGA	*Strip packing*	Discreta
Martinez-Sykora *et al.* (2017)	*Math*-heurística	*Bin packing*	Discreta
Mundim *et al.* (2018)	H4NP	*Placement, cutting stock, knapsack* e *bin packing*	Discreta
Sato *et al.* (2019)	ROMA	*Strip packing*	Discreta
Souza Queiroz e Andretta (2020)	BRKGA	*Two-dimensional irregular knapsack*	Discreta
Chehrazad, Roose e Wauters (2022)	*Bottom-left* semi-discreto	*Strip packing*	Discreta
Zhang *et al.* (2022)	Busca local	*Bin packing*	Discreta
Umetani e Murakami (2022)	GCDH	*Strip packing*	Discreta
Lu, Hu e Ng (2023)	BRKGA	*Strip packing*	Discreta
Sato *et al.* (2023)	FOMLS	*Strip packing* com uma e duas dimensões abertas, e com objeto quadrado	Discreta

vantajoso, uma vez que: i) podem ser gerados muitos pontos desnecessários, o que aumenta o custo computacional; e ii) alguns itens podem ter poucas posições adequadas, o que prejudica os encaixes dos itens e, consequentemente, a qualidade da solução. Para mitigar estes problemas, os autores propõem diferentes formas de gerar malhas.

Outro trabalho que aprimora o modelo foi o de Rodrigues e Toledo (2017), que substituem as restrições de não-sobreposição (2.20) por restrições baseadas em cliques e propõem uma nova forma de calcular um limitante dual. Os resultados mostraram que o modelo modificado apresenta melhores resultados para a maioria dos casos tanto no tempo de resolução quanto no uso de memória (houve instâncias em que não foi possível gerar o modelo de Toledo *et al.* (2013) devido ao grande número de restrições). Entretanto, neste trabalho é utilizado o modelo de Toledo *et al.* (2013) sem essas modificações, pois desta forma a adaptação para o problema estudado é mais simples.

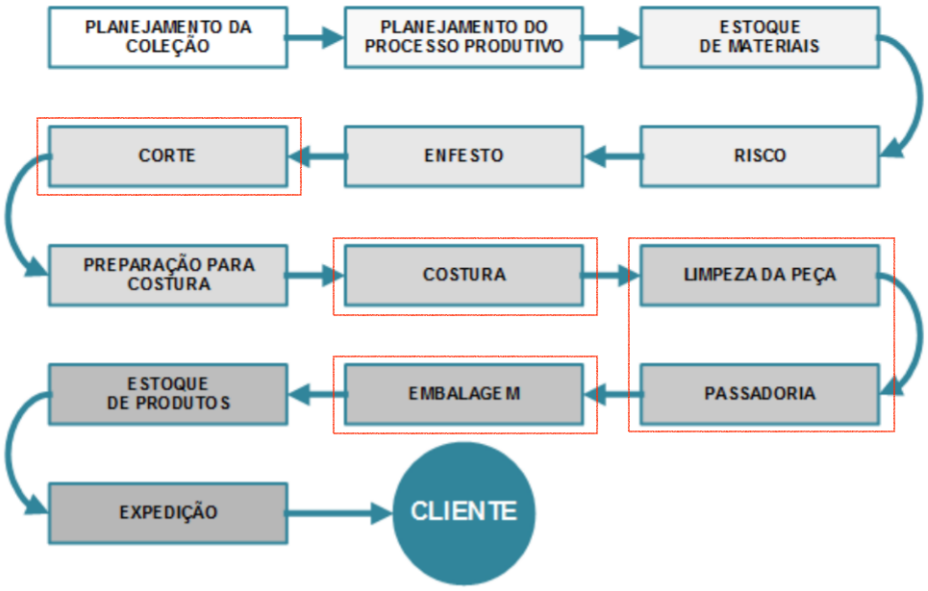
CAPÍTULO

3

DEFINIÇÃO DO PROBLEMA E MODELAGEM MATEMÁTICA[1]

[1]De acordo com Oliveira e Todaro (2014), o processo de produção de roupas pode ser descrito de forma simplificada como ilustrado na Figura 16. De forma ainda mais simplificada, o processo pode ser dividido em quatro etapas principais: corte, costura, acabamento e embalagem. Estas etapas estão destacadas na Figura 16 pelos retângulos vermelhos.

Figura 16 – Processo de confecção simplificado.



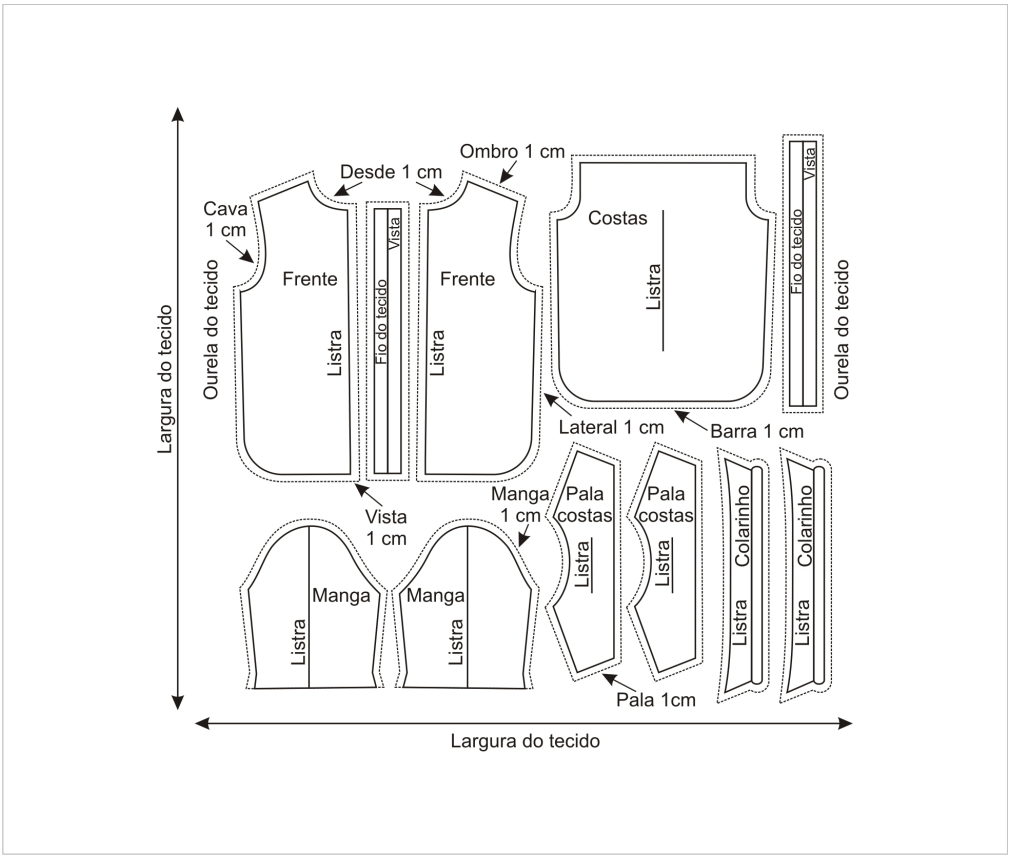
Fonte: Adaptado de Oliveira e Todaro (2014).

Na etapa de corte, a demanda das peças de roupas é recebida e os moldes das roupas são separados. Os moldes podem ser compostos por diferentes peças (itens). Em seguida, inicia-se o processo de corte propriamente dito que em confecções, em geral, é realizado em mesas, onde o

[1] Parte dos resultados deste capítulo foram publicados em Hono e Toledo (2023).

tecido é disposto em camadas formando um retângulo (chamado, na literatura, de objeto ou *bin*). O objeto tem largura fixa (determinada pela largura da mesa de corte) enquanto seu comprimento pode variar entre um mínimo e um máximo. A partir dos itens demandados, um ou mais planos de corte são elaborados. Um plano de corte estabelece a posição em que cada item é alocado no objeto para melhor aproveitar o tecido, visando reduzir a perda de matéria-prima. Na Figura 17, é ilustrado um plano de corte de uma camisa. Note que cada camisa (produto final) é formada por um conjunto de itens (p.e., frente, costas, mangas, colarinho e punho).

Figura 17 – Exemplo de solução para o corte de um modelo de camisa.



Fonte: Tecnologia-Treinamento (2024).

Neste trabalho, é estudada a etapa de corte, que como observado no fluxograma, é a etapa inicial do processo de produção de uma confecção e impacta diretamente em suas demais etapas (Alsamarah, Younes e Yousef (2022)). Mais especificamente, será abordada a elaboração de um ou mais planos de corte que, do ponto de vista de otimização, pode ser representada como um problema de empacotamento de peças irregulares em *bin*. Este trabalho estudará o problema de corte de produtos finais (roupas) que são compostos por diferentes peças (itens), isto é, é necessário manter num mesmo plano de corte todos os itens de um mesmo produto final. Assim, o objetivo é minimizar a quantidade de *bins* (objetos de corte) necessários para empacotar todos os produtos finais demandados, enquanto é assegurado que todos os itens que formam um mesmo produto final sejam cortados a partir de um mesmo *bin*. Este tipo de restrição é conhecido como *complete-shipment constraints* (Bischoff e Ratcliff (1995) e Bortfeldt e Wäscher (2013)). Esta

questão é relevante, em especial, por dois motivos: a) permitir que ao término do corte de um objeto, o processo de costura dos produtos finais possa ser iniciado (desta forma evitando um gargalo na cadeia de produção); e b) assegurar a padronização dos produtos finais, evitando possíveis diferenças de tonalidade no tecido dos itens que compõem um mesmo produto, como explica M'Hallah e Bouziri (2016). Para evitar a produção em excesso de um mesmo produto final, também é imposta uma demanda máxima para cada produto, assim criando um estoque balanceado de produtos.

Além dessas restrições, há as restrições do problema que devem garantir que todas as peças estejam completamente contidas no objeto e que elas não se sobreponham. A altura e comprimento do *bin* são limitados pelo tamanho da mesa de corte, entretanto é permitida uma redução no comprimento do *bin*, pois, no caso de não ser possível empacotar nenhum produto final a mais nele, reduzir o comprimento do objeto faz com que menos matéria-prima seja desperdiçada. Porém, vale ressaltar que grandes reduções no comprimento prejudicam a eficiência da produção, pois isso faz com que uma menor parte da mesa de corte seja utilizada. Outro ponto importante do problema estudado é que rotações das peças não são permitidas, pois o tecido possui uma orientação das fibras e algumas peças de roupas podem ser listradas (como em Alves (2016)) ou outras estampas. Assim, é necessário manter um padrão nas peças para a garantia da qualidade do produto final.

Na literatura, M'Hallah e Bouziri (2016) estudaram o mesmo problema, porém considerando também quantas camadas de tecido deveriam ser colocadas em cada objeto de corte. Entretanto, os autores não consideraram penalização no caso em que a demanda era ultrapassada. Os autores propuseram três heurísticas para o problema, sendo duas delas algoritmos genéticos, enquanto a terceira se baseava na meta-heurística *simulated annealing* (SA).

Entretanto, diferentemente de M'Hallah e Bouziri (2016) que utilizou um método heurístico, neste capítulo será apresentado um modelo matemático o problema. Na Seção 3.1 é apresentado o modelo desenvolvido. Na Seção 3.2 são mostrados os experimentos realizados com o modelo e seus resultados. A Seção 3.3 faz um levantamento dos resultados obtidos.

3.1 Modelagem matemática

O problema de empacotamento em *bins* como aqui estudado foi modelado tendo como base o modelo de Toledo *et al.* (2013). Este modelo discretiza o espaço dentro do objeto de corte com uma malha regular e permite o posicionamento dos itens apenas nestas coordenadas. Devido à grande complexidade computacional do problema (pois é NP-difícil como explica Fowler, Paterson e Tanimoto (1981)), inicialmente foi considerado o empacotamento de roupas em apenas um *bin*, isto é, produtos completos ou finais com objetivo de maximizar a área aproveitada do *bin*, ou seja, minimizar a perda de matéria-prima. O *bin* tem um comprimento mínimo e máximo que são definidos de acordo com o tamanho da mesa de corte. Os conjuntos,

os parâmetros e as variáveis do modelo são apresentados abaixo. Em seguida, são destacadas as adaptações realizadas em relação ao modelo de Toledo *et al.* (2013).

Conjuntos

- W é o conjunto de tipos de roupas a serem confeccionadas (produtos finais);
- I é o conjunto de tipos de itens, em que $i \in I$ é o item do tipo i no conjunto;
- $\mathscr{B}$ é o conjunto de todos os pontos da malha;
- $\mathscr{B}_i \subset \mathscr{B}$ é o conjunto de pontos para os quais a alocação do item do tipo i é factível, ou seja, o item fica inteiramente contido no objeto (*inner-fit-polygon*);
- NFP_{ij}^k é o subconjunto de pontos de $\mathscr{B}$ que compõe o *no-fit polygon* entre os itens de tipo i e j caso o item i seja alocado no ponto k. Ou seja, este conjunto de pontos indica que se o item j for posicionado $p \in NFP_{ij}^k$, então haverá sobreposição entre os itens i e j.

Parâmetros

- L é largura do objeto de corte;
- x_k é a coordenada x do ponto $k \in \mathscr{B}$;
- c_i é a diferença entre a coordenada x do ponto de referência do item i e a coordenada x do vértice mais à direita deste mesmo item;
- p_w é a área ocupada pelo produto final $w \in W$;
- $d_w^{\min}$ e $d_w^{\max}$ são os limitantes inferiores e superiores para a demanda de cada tipo de produto final $w \in W$;
- n_{wi} é o número de itens do tipo $i \in I$ necessários para a produção de um produto final $w \in W$;
- Min_c e Max_c são o comprimento mínimo e máximo permitidos para o *bin*, respectivamente.

Variáveis

- Δ_w é uma variável inteira que corresponde ao número de produtos finais do tipo $w \in W$ cortados;
- δ_{ik} é uma variável binária que assume o valor 1 se um item do tipo i é alocado no ponto k e assume 0 caso contrário;
- z é o comprimento total utilizado.

$$\min \quad L\,z - \sum_{w \in W} p_w \Delta_w \tag{3.1}$$

s.a

$$\delta_{jt} + \delta_{ik} \leq 1, \qquad \forall t \in NFP^k_{ij};\ \forall i, j \in I,\ i \geq j;\ \forall k \in \mathscr{B}_i, \tag{3.2}$$

$$d_w^{\min} \leq \Delta_w \leq d_w^{\max}, \qquad \forall\, w \in W, \tag{3.3}$$

$$\sum_{w \in W} n_{wi}\,\Delta_w = \sum_{k \in B_i} \delta_{ik}\ , \quad \forall i \in I, \tag{3.4}$$

$$(x_k + c_i)\ \delta_{ik} \leq z, \qquad \forall\ i \in I,\ k \in \mathscr{B}_i, \tag{3.5}$$

$$Min_c \leq z \leq Max_c, \tag{3.6}$$

$$\Delta_w \in \mathbb{N}, \qquad \forall w \in W, \tag{3.7}$$

$$\delta_{ij} \in \{0,1\}, \qquad \forall\ i \in I,\ j \in \mathscr{B}_i. \tag{3.8}$$

O modelo busca uma solução com menor desperdício de matéria-prima (3.1), ou seja, minimiza a diferença entre área do objeto menos a soma das áreas das peças cortadas. A factibilidade do empacotamento dos itens no objeto é representada pelo conjunto de restrições (3.2), em que dois itens não podem se sobrepor e devem ser alocados de forma a estarem completamente contidos no objeto ($k \in \mathscr{B}_i$). As restrições (3.3) estabelecem os limites inferior $d_w^{\min}$ e superior $d_w^{\max}$ de produção para cada produto. As restrições (3.4) estabelecem a correspondência entre o número de produtos finais cortados e os itens que ele demanda, desta forma é assegurado que cada produto final tem todos os itens disponíveis para sua produção. As restrições (3.5) relacionam o comprimento do objeto com as peças posicionadas. A restrição (3.6) limita o comprimento mínimo e máximo do objeto. O domínio das variáveis é definido em (3.7) e (3.8).

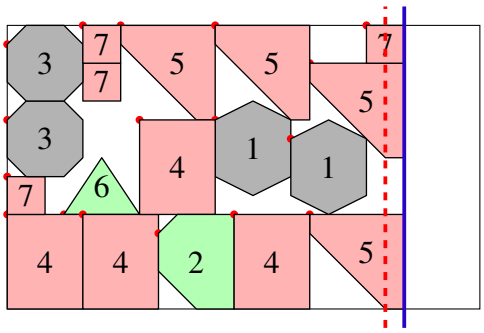
As restrições (3.4) e (3.6) foram adicionadas ao modelo de Toledo *et al.* (2013) para tratar o corte de produtos finais e sua função objetivo foi alterada para representar a perda de matéria-prima.

Na Figura 18, é ilustrado um exemplo de solução para o problema. Neste exemplo, os itens de mesma cor representam partes de um mesmo produto final (ou seja, os conjuntos {1, 3}, {2, 6} e {4, 5, 7} formam produtos finais), os número correspondem ao tipo de item e os pontos vermelhos indicam seus pontos de referência. A linha vermelha pontilhada representa o comprimento mínimo admitido pelo *bin* e a linha azul o comprimento do *bin* utilizado na solução encontrada.

3.1.1 Adaptação para múltiplos objetos de corte

Em uma confecção, os produtos finais são demandados em quantidade. Isto resulta no planejamento de vários planos de corte. Como mencionado anteriormente, estes planos deveriam ser definidos considerando o corte de produtos finais, pois assim tanto o processo de costura pode ser iniciado na sequência quanto a qualidade dos produtos finais é uniforme.

Figura 18 – Exemplo de empacotamento de produtos finais.



Fonte: Elaborado pelo autor.

Na seção anterior, foi considerado o corte de produtos finais para um plano de corte, logo para atender a demanda de uma confecção, o modelo deveria ser utilizado de forma hierárquica, ou seja, uma vez obtido um plano de corte, é reduzida a demanda dos produtos cortados e um novo plano é definido. No entanto, esta estratégia pode levar a uma aproximação da solução ótima do problema. A estratégia ótima deveria considerar a definição de todos os planos de corte em conjunto (estratégia integrada), logo o modelo deveria tratar simultaneamente vários planos de corte.

Para tratar o problema de empacotamento em *bins*, é possível modificar o modelo da Seção 3.1 adicionando um índice a todas as variáveis. Este novo índice representa o *bin* em que os itens estão. Além disso, também é necessário adicionar uma variável binária que indica se o *bin* é utilizado ou não. Sendo $\mathscr{O} = \{1, ..., N\}$ o conjunto de *bins* disponíveis, o modelo poderia ser reescrito como definido a seguir:

$$\min \quad \sum_{r\in\mathscr{O}} (L\, z^r - \sum_{w\in W} p_w \Delta_w^r) \tag{3.9}$$

s.a

$$\delta_{jt}^r + \delta_{ik}^r \leq 1, \qquad \forall t \in NFP_{ij}^k;\ \forall i, j \in I,\ i \geq j;\ \forall k \in \mathscr{B}_i;\ \forall r \in \mathscr{O}, \tag{3.10}$$

$$d_w^{\min} \leq \sum_{r\in\mathscr{O}} \Delta_w^r \leq d_w^{\max}, \qquad \forall\, w \in W, \tag{3.11}$$

$$\sum_{w\in W} n_{wi}\, \Delta_w^r = \sum_{k\in B_i} \delta_{ik}^r, \qquad \forall i \in I;\ \forall r \in \mathscr{O}, \tag{3.12}$$

$$(x_k + c_i)\ \delta_{ik}^r \leq z^r, \qquad \forall\, i \in I;\ k \in \mathscr{B}_i;\ \forall r \in \mathscr{O}, \tag{3.13}$$

$$Min_c\ y^r \leq z^r \leq Max_c\ y^r, \qquad \forall r \in \mathscr{O}, \tag{3.14}$$

$$y^r \geq\ y^{r+1}, \qquad \forall r \in \mathscr{O}. \tag{3.15}$$

$$\Delta_w^r \in Z^+ \qquad \forall w \in W,\ \forall r \in \mathscr{R} \tag{3.16}$$

$$\delta_{ik}^r \in \{0,1\} \qquad \forall\, i \in I,\ k \in \mathscr{B}_i;\ \forall r \in \mathscr{O} \tag{3.17}$$

$$y^r \in \{0,1\} \qquad \forall r \in \mathscr{O}. \tag{3.18}$$

Desta forma, a função objetivo (3.9) minimiza a soma dos desperdícios dos *bins*. A nova variável binária y^r é igual a 1 se o *bin* r é utilizado e 0 caso contrário. As restrições (3.14) impõem que o uso do *bin* r seja permitido apenas se $y^r = 1$ e também que z^r esteja dentro do intervalo permitido para o comprimento do *bin*. As restrições (3.15) eliminam soluções simétricas, pois permitem o uso de um próximo *bin* apenas se o *bin* anterior já estiver em uso. As demais restrições são análogas ao modelo anterior. O domínio das variáveis é definido em (3.16), (3.17) e (3.18).

3.2 Experimentos computacionais

Para analisar o estudo do problema de empacotamento de produtos finais, foram realizados experimentos computacionais divididos em quatro fases. Inicialmente, o objetivo foi avaliar a relevância de empacotar os itens considerando os produtos finais ao invés de empacotar apenas os itens individualmente. Nesta fase, foi considerado o empacotamento em apenas um *bin*. Em seguida, experimentos computacionais foram realizados considerando o empacotamento em múltiplos *bins*. Na terceira fase, é estudado o impacto que o comprimento mínimo permitido do objeto de corte causa na solução. A quarta fase realiza os mesmos testes que a terceira fase, porém utilizando outra malha de pontos. Mais especificamente, foi utilizada a malha *piece based mesh* proposta por Cherri *et al.* (2018).

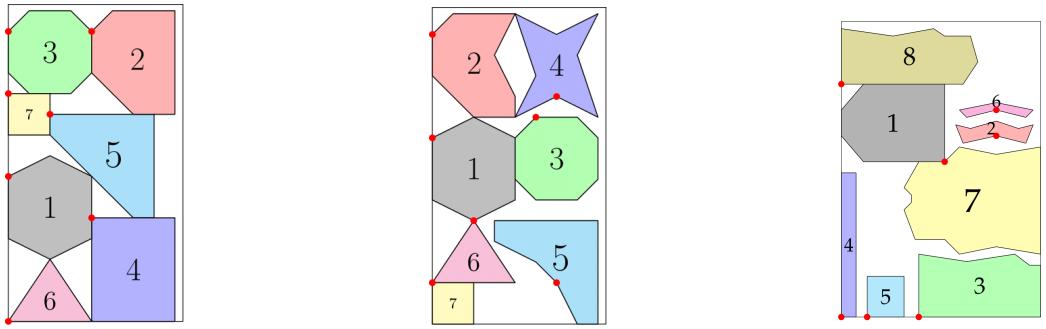
Os modelos desenvolvidos foram codificados em linguagem Julia (versão 1.7.3) utilizando a biblioteca JuMP. Para resolver as instâncias-teste foi utilizado o solver Gurobi (v9.5.1). Os testes foram realizados em um computador com processador Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz, 16Gb de memória RAM e sistema operacional Ubuntu 22.04.2 LTS (64 bits). O tempo-limite para a primeira fase foi de 1800 segundos; para as demais fases, o tempo limite foi de 7200 segundos, pois as instâncias destas fase possuem dimensões maiores.

3.2.1 Instâncias-teste

As instâncias utilizadas foram baseadas nas instâncias *rco*, *blazewicz* e *marques* disponíveis em ESICUP (2021). A Figura 19 ilustra os itens dessas instâncias. O ponto em vermelho indica o ponto de referência para posicionar os itens. Vale ressaltar que os itens da instância *rco* são a envoltória convexa dos itens da instância *blazewicz*, então essas instâncias servem para avaliar a sensibilidade do modelo em relação à irregularidade dos itens. Já a instância *marques* é baseada em um plano de corte da indústria têxtil, portanto é uma instância próxima ao problema real. Os itens do tipo 1, 6 e 7 da instância *marques* foram rotacionados em 90° (sentido anti-horário) para que ficassem mais próximos de um plano de corte real.

As instâncias são descritas na Tabela 3 em que a coluna "Instância" identifica a instância, as colunas "L" e "Max_c" são, respectivamente, a largura e comprimento máximos do objeto. O valor p que procede o nome da instância é a porcentagem de redução do tamanho do *bin*

Figura 19 – Itens das instâncias *rco*, *blazewicz* e *marques*.



(a) *rco*. (b) *blazewicz*. (c) *marques*.

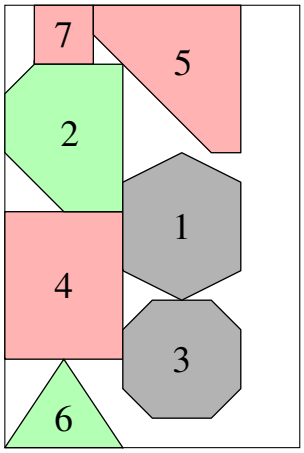
Fonte: Retirado de Hono e Toledo (2023).

permitida (por exemplo, se $p = 0,5$ e $Max_c = 10$, segue que $Min_c = 5$). A coluna "Produtos Finais" se refere aos conjuntos em que os itens devem ser empacotadas, ou seja, os conjuntos que equivalem a cada produto final. Por exemplo, na instância rco_2_p, o primeiro produto final é composto pelos itens 1 e 3, o segundo produto final é composto pelos itens 4, 5 e 7 e o terceiro produto final é composto pelos itens 2 e 6. Na Figura 20, são ilustrados estes produtos finais citados, em que itens da mesma cor indicam que fazem parte do mesmo produto. No caso da instância $marques_2$, o número entre parênteses indica a quantidade do item que o produto final deve ter. É importante destacar que a primeira instância de cada conjunto (*rco*, *blazewicz* e *marques*) representa o empacotamento sem considerar produtos finais (irrestrito).

Tabela 3 – Dados das instâncias utilizadas.

Instância	L	Max_c	Produtos Finais
rco_1_p	15	25	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$
rco_2_p	15	25	$\{1,3\}, \{4,5,7\}, \{2,6\}$
rco_3_p	15	25	$\{1,5\}, \{2,7\}, \{3,4,6\}$
rco_4_p	15	25	$\{1,6,7\}, \{2,4\}, \{3,5\}$
rco_5_p	15	25	$\{1,2,7\}, \{3,6\}, \{4,5\}$
$blazewicz_1\_p$	15	25	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}$
$blazewicz_2\_p$	15	25	$\{1,3\}, \{4,5,7\}, \{2,6\}$
$blazewicz_3\_p$	15	25	$\{1,5\}, \{2,7\}, \{3,4,6\}$
$blazewicz_4\_p$	15	25	$\{1,6,7\}, \{2,4\}, \{3,5\}$
$blazewicz_5\_p$	15	25	$\{1,2,7\}, \{3,6\}, \{4,5\}$
$marques_1\_p$	104	340	$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}$
$marques_2\_p$	104	340	$\{1(2),2(2),3(1),4(1),5(2),6(2),7(1),8(1)\}$

Figura 20 – Produtos da instância rco_2_p.



Para representar as instâncias-teste nas três primeiras fases, foi utilizado o modelo matemático considerando uma malha de pontos regular com discretização unitária para as instâncias do tipo *rco* e *blazewicz*. Para as instâncias do tipo *marques* foi considerada uma discretização de 7x7 devido à dimensão do objeto de corte. Para a quarta fase, foi utilizada uma malha alternativa proposta por Cherri *et al.* (2018), que tem como proposta gerar malhas de pontos específicas para cada tipo de item com base em suas dimensões.

3.2.2 Fase I - Empacotamento de produtos finais

Como descrito no Capítulo 3, é importante que produtos finais sejam empacotados em conjunto dentro de cada *bin* para assegurar que não exista diferença na tonalidade de cor das partes que os compõem e também possibilitar que a etapa de costura seja iniciada logo após o término da etapa de corte.

Sendo assim, os experimentos desta fase comparam a quantidade de produtos finais obtidos ao impor ou não que os itens sejam empacotados de acordo com seus respectivos produtos finais. Nesta fase, o tempo limite de execução foi de 30 minutos (1800 segundos) e foi utilizado $p = 0,9$, pois representa a possibilidade uma pequena redução no comprimento do *bin*. Não foi imposta demanda para os produtos (ou seja, não foram utilizadas as restrições (3.3) ou (3.11)), pois o objetivo é apenas maximizar a ocupação do *bin*.

Figura 21 – Plano de corte - Instância rco_1_90.

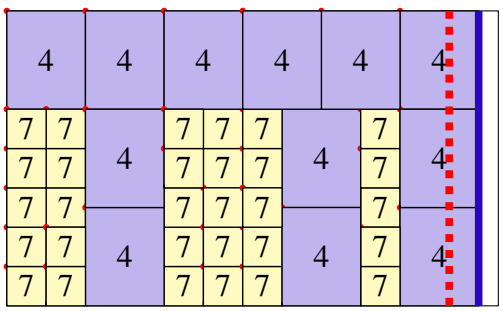


Tabela 4 – Soluções obtidas utilizando a abordagem desenvolvida.

Instância	Min_c	z	Desp. Total	GAP (%)	T (s)	Δ_{total}	Produtos Finais
rco_1_90	22,5	24,0	0,0	0,0	13	-	0, 0, 0, 12, 0, 0, 30
rco_2_90	22,5	23,0	31,4	0,0	921	9	2 (0),5 (0),2 (0)
rco_3_90	22,5	24,0	32,6	18,4	1.800	12	1 (0),8 (0),3 (0)
rco_4_90	22,5	24,0	22,5	0,0	75	9	0 (0),9 (0),0 (0)
rco_5_90	22,5	24,0	36,1	8,0	1.800	10	5 (0),3 (0),2 (0)
$blazewicz_1$_90	22,5	24,0	18,0	0,0	756	-	0, 0, 1, 0, 4, 5, 59
$blazewicz_2$_90	22,5	23,0	65,8	43,9	1.800	10	2 (0),4 (0),4 (0)
$blazewicz_3$_90	22,5	23,0	57,0	44,8	1.800	11	2 (0),4 (0),5 (0)
$blazewicz_4$_90	22,5	25,0	62,2	26,7	1.800	11	7 (0),0 (0),4 (1)
$blazewicz_5$_90	22,5	23,0	60,0	34,6	1.800	11	2 (0),5 (1),4 (0)
$marques_1$_90	306,0	313,0	3906,0	0,0	1.242	-	44, 0, 9, 2, 0, 0, 0, 0
$marques_2$_90	306,0	334,0	10322,1	27,9	1.800	7	7 (0)

A Tabela 4 resume os resultados obtidos. A coluna “Instância” identifica a instância resolvida, “Min_c” é o comprimento mínimo que o *bin* pode assumir, “z” é o comprimento do *bin* na melhor solução obtida, “Desp. Total” é a somatória dos desperdícios de matéria-prima dos *bins* em unidade de área (note que nessa fase é considerado apenas um *bin*, logo “Desp. Total” é o desperdício deste único *bin*), “GAP (%)” é o desvio relativo percentual entre a melhor solução obtida e seu limitante dual, “T (s)” é o tempo (em segundos) de resolução da instância e “Δ_{total}” é a quantidade total de itens empacotados. Note que não há valores no caso das instâncias de empacotamento irrestrito, pois elas não possuem produtos finais a serem confeccionados. Na coluna “Produtos Finais”, são reportadas as quantidade de cada produto final cortados, os números entre parênteses indicam a quantidade que seria obtida se fosse considerado apenas o empacotamento individual das peças (ou seja, a primeira instância de cada tipo). Por exemplo, a Figura 21 ilustra o plano de corte obtido para a instância rco_1_90 (ou seja, doze itens do tipo 4 e trinta itens do tipo 7 são cortados, a linha azul indica o comprimento utilizado do *bin* e a linha vermelha indica o comprimento mínimo permitido para o *bin*). Dessa forma, utilizando este plano como base, note que não seria possível obter nenhum produto final ao considerar qualquer outra instância de empacotamento em conjunto (rco_2_90, rco_3_90, rco_3_90 e rco_4_90), pois nenhum produto final utiliza apenas os itens do tipo 4 e 7. Portanto, os números em parênteses para estas instâncias são zero.

Os resultados mostram que considerar o empacotamento em conjunto é relevante, pois, caso contrário, não seria possível obter produtos finais na maioria dos casos, pois note que, com exceção das instâncias $blazewicz_4$_90 e $blazewicz_5$_90, nas demais instâncias não foi possível obter qualquer produto final ao considerar o plano de empacotamento irrestrito. Além disso, mesmo que as instâncias $blazewicz_4$_90 e $blazewicz_5$_90 consigam gerar produtos finais utilizando o plano de corte irrestrito, o número de produtos gerados foi baixo comparado com os planos de empacotamento em conjunto. Isso atrasaria a linha de produção, uma vez que não é possível iniciar o processo de costura até que todas as partes que compõem um produto sejam

obtidas. Portanto, mesmo que os planos de cortes irrestritos apresentem menor desperdício de matéria-prima, ainda é preferível utilizar os planos que consideram os produtos finais.

3.2.3 Fase II - Estudo do modelo com múltiplos objetos

A segunda fase de testes estuda o desempenho do modelo para instâncias-teste com múltiplos *bins*. As demandas mínimas e máximas impostas para as instâncias *rco* e *blazewicz* foram 1 e 8, respectivamente. Para a instância do tipo *marques*, as demandas mínima e máxima foram 1 e 12. A quantidade de *bins* variou de 1 até 3 para as instâncias do tipo *rco* e *blazewicz*, enquanto a instância *marques* considerou apenas 1 *bin*, pois esta instância possui apenas um produto final e, portanto, a solução ótima considerando mais de um objeto de corte pode ser obtida repetindo-se o melhor plano de corte encontrado considerando um único *bin*. Além disso, é imposto que todos os *bins* sejam utilizados, ou seja, $y^r = 1,\ \forall r \in \mathscr{O}$. O comprimento mínimo imposto foi 90% ($p = 0,9$). Como este é um problema de maior dimensão, foi considerado o tempo limite de execução de 2 horas (7200 segundos).

Na Tabela 5, são reportados os resultado obtidos. A coluna "Nb" indica o número de *bins* considerados, "z^r" representa o comprimento utilizado de cada *bin* na melhor solução obtida, respectivamente, "Δ_w^r" indica a quantidade de cada produto final obtida em cada *bin* (cada colchetes indica um *bin* e os números dentro deles é a quantidade de produtos finais produzidos de cada tipo) e "DpP" é o desperdício por produto final, ou seja, a razão dos desperdício total e a quantidade de produtos finais. As demais colunas apontam as mesmas informações descritas na Tabela 4.

Os resultados mostraram que resolver o problema considerando múltiplos objetos de corte é mais difícil devido à dimensão do problema. Não foi possível provar a otimalidade em nenhuma instância com dois *bins* e o GAP dessas soluções sempre é maior em relação à solução para apenas um *bin*. Entretanto, as instâncias com 3 objetos de corte são resolvidas sem dificuldades, pois o espaço mínimo (considerando o menor comprimento permitido) disponível em todos os *bins* é o suficiente para empacotar com folga a demanda máxima dos produtos finais.

Analisando a qualidade dos planos de corte, conclui-se que utilizar apenas um *bin* é mais vantajoso tanto em relação ao desperdício de matéria-prima quanto no custo-benefício (desperdício por produto final). O desperdício aumenta conforme mais objetos de corte são considerados, pois são somados as perdas de todos os planos de corte. Assim, ao posicionar mais itens, é sempre perdida uma determinada quantidade de matéria-prima, com exceção do caso em que os itens se encaixam perfeitamente.

Entretanto, o desperdício por produto final dos planos de corte que consideram dois *bins* são próximos dos planos com um objeto de corte, tendo potencial de serem superiores caso o tempo limite seja aumentado. Em relação aos planos com três *bins*, é observado que o desperdício total e o desperdício por produto final são constantes para cada tipo de instância,

Tabela 5 – Resultados computacionais para múltiplos objetos.

Instância	Nb	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	GAP (%)	T (s)
rco_2_90	1	22,5	[23,0]	31,4	[2,5,2]	9	3,5	0,0	1030
rco_3_90		22,5	[24,0]	32,6	[1,8,3]	12	2,7	0,0	2185
rco_4_90		22,5	[24,0]	34,3	[4,5,1]	10	3,4	0,0	1544
rco_5_90		22,5	[24,0]	36,1	[5,3,2]	10	3,6	0,0	2621
rco_2_90	2	22,5	[25,0; 23,0]	84,8	[4,4,2], [3,4,2]	19	4,5	41,3	7200
rco_3_90		22,5	[23,0; 25,0]	85,9	[4,2,3], [2,5,4]	20	4,3	44,4	7200
rco_4_90		22,5	[23,0; 23,0]	78,4	[5,2,3], [2,6,1]	19	4,1	36,0	7200
rco_5_90		22,5	[24,0; 23,0]	74,4	[4,3,3], [4,4,2]	20	3,7	26,2	7200
rco_2_90	3	22,5	[22,5; 22,5; 22,5]	250,1	[3,2,3], [2,3,3], [3,3,2]	24	10,4	0,0	42
rco_3_90		22,5	[22,5; 22,5; 22,5]	250,1	[2,3,3], [3,3,2], [3,2,3]	24	10,4	0,0	42
rco_4_90		22,5	[22,5; 22,5; 22,5]	250,1	[3,3,2], [3,2,3], [2,3,3]	24	10,4	0,0	65
rco_5_90		22,5	[22,5; 22,5; 22,5]	250,1	[4,2,2], [2,4,2], [2,2,4]	24	10,4	0,0	16
$blazewicz_2$_90	1	22,5	[24,0]	55,8	[5,2,4]	11	5,1	24,7	7200
$blazewicz_3$_90		22,5	[24,0]	55,5	[3,2,5]	10	5,6	24,5	7200
$blazewicz_4$_90		22,5	[24,0]	53,7	[6,3,2]	11	4,9	39,3	7200
$blazewicz_5$_90		22,5	[23,0]	55,1	[4,3,3]	10	5,5	31,8	7200
$blazewicz_2$_90	2	22,5	[23,0; 22,5]	133,5	[3,3,4], [4,4,1]	19	7,0	47,5	7200
$blazewicz_3$_90		22,5	[23,0; 23,0]	126,9	[4,3,3], [2,4,4]	20	6,3	44,7	7200
$blazewicz_4$_90		22,5	[23,0; 23,0]	124,2	[3,3,4], [3,3,4]	20	6,2	45,3	7200
$blazewicz_5$_90		22,5	[23,0; 23,0]	124,9	[3,3,4], [4,4,2]	20	6,2	45,6	7200
$blazewicz_2$_90	3	22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [2,3,3], [4,2,2]	24	13,9	0,0	24
$blazewicz_3$_90		22,5	[22,5; 22,5; 22,5]	334,1	[3,2,3], [2,3,3], [3,3,2]	24	13,9	0,0	44
$blazewicz_4$_90		22,5	[22,5; 22,5; 22,5]	334,1	[1,3,4], [4,3,1], [3,2,3]	24	13,9	0,0	49
$blazewicz_5$_90		22,5	[22,5; 22,5; 22,5]	334,1	[3,3,2], [3,2,3], [2,3,3]	24	13,9	0,0	40
$marques_2$_90	1	306,0	[332,0]	10114,1	[7]	7	1444,9	21,6	7200

pois é possível alocar todos os produtos finais utilizando o comprimento mínimo do *bin*.

Um ponto importante é que na maioria das soluções encontrada, não foi utilizado o comprimento máximo do objeto de corte. Isso indica que permitir uma pequena redução no *bin* ajuda a diminuir o desperdício total, pois em alguns casos, não é possível alocar mais produtos finais no plano de corte, porém ainda há espaço livre no objeto de corte. Assim, permitir essa redução ajuda a gerar um plano de corte mais compacto.

3.2.4 Fase III - Estudo sobre o comprimento mínimo do objeto

A terceira fase de experimentos analisa o impacto que diferentes comprimentos mínimos nos *bins* causam na solução final. Os testes anteriores (Tabela 5) mostraram que permitir o uso mínimo de 90% do comprimento melhora o desperdício nos planos de corte, uma vez que a maioria das soluções não utilizaram o comprimento máximo do objeto de corte (observe que apenas as instâncias rco_2_90 e rco_3_90 apresentaram planos de corte que utilizam o comprimento máximo do *bin*). Entretanto, planos de corte menores geram uma perda de produtividade, pois uma parte menor da mesa de corte é utilizada.

Sendo assim, foram considerados comprimentos mínimos de 50%, 60%, 70%, 80% e 90% do comprimento do objeto ($p \in \{0,5;\ 0,6;\ 0,7;\ 0,8;\ 0,9\}$). O tempo limite de execução foi de 2 horas (7200 segundos).

Nas Tabelas 6, 7 e 8 são reportados os resultados obtidos, enquanto, na Figura 22, é ilustrada a solução obtida para a instância $marques_2$_90 (a linha azul vertical indica o comprimento

utilizado do objeto e a linha vermelha tracejada indica o comprimento mínimo permitido).

Em todas as instâncias, o aumento no comprimento mínimo causou um aumento no desperdício total de matéria-prima. Isso se deve ao fato de que não é possível encaixar os itens perfeitamente e, portanto, ao alocar mais itens, também é gerado um maior desperdício. Para as instâncias do tipo *rco* e *blazewicz*, é observado que um aumento no comprimento mínimo leva a planos de corte com maior custo benefício na maioria dos casos (ou seja, diminui o desperdício por produto final). Vale destacar que, considerando as instâncias com três objetos de corte, são observados casos em que todos os produtos finais podem ser empacotados com folga e, neste caso, ocorre um grande aumento no desperdício por produto final. Em relação às instância do tipo *marques*, nota-se que os melhores desperdício por produto final são obtidos por $p = 0,5$ e $p = 0,9$, porém o plano obtido por $p = 0,9$ é preferível, uma vez que faz um maior uso da mesa de corte.

Também é válido destacar que o modelo evita empacotar um grande número de produtos finais, pois isso aumenta o desperdício total calculado pela função objetivo. Isso mostra que a função objetivo precisa ser modificada para que bonifique também o custo-benefício de um plano de corte com muitos produtos finais. Uma alternativa para isso seria utilizar uma função multi-objetivo que penalize o desperdício total, enquanto bonifique a quantidade de produtos empacotados.

Figura 22 – Solução obtida para a instância *marques*$_2$_90.

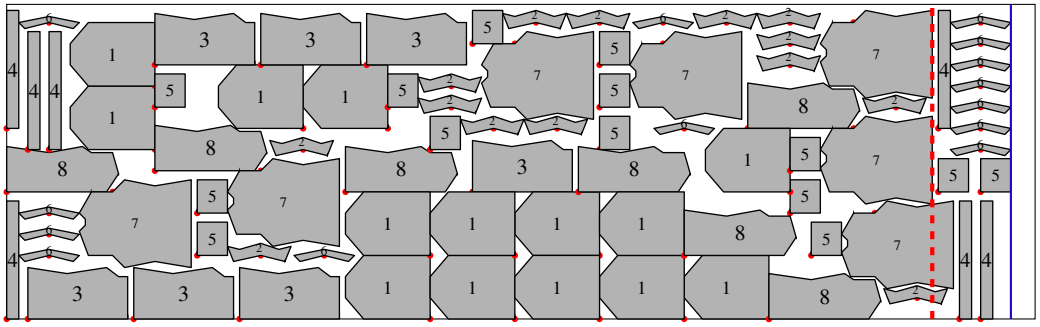
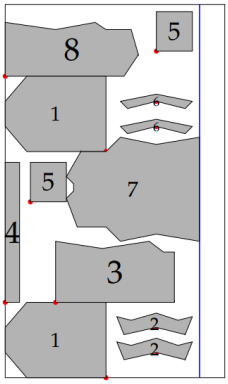
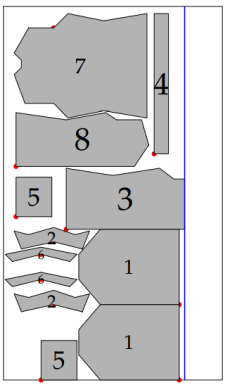


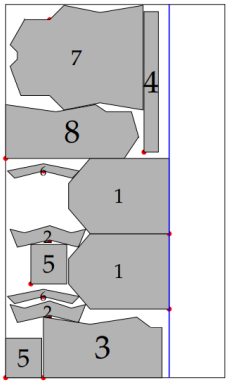
Figura 23 – Soluções obtidas para a instância *marques* com um único produto e três malhas.



(a) *marques*$_2$_0_7,0.



(b) *marques*$_2$_0_3,5.



(c) *marques*$_2$_0_1,75.

Tabela 6 – Resultados obtidos variando o comprimento mínimo - Instância *rco*.

Instância	Nb	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	GAP (%)	T (s)
rco_2_50		12,5	[15,0]	23,4	[1,3,2]	6	3,9	0,0	1.111
rco_2_60		15,0	[15,0]	23,4	[1,3,2]	6	3,9	0,0	437
rco_2_70	1	17,5	[19,0]	25,7	[1,5,1]	7	3,7	0,0	983
rco_2_80		20,0	[21,0]	31,4	[1,5,2]	8	3,9	0,0	1.164
rco_2_90		22,5	[23,0]	31,4	[2,5,2]	9	3,5	0,0	777
rco_2_50		12,5	[13,0; 13,0]	52,1	[2,2,1], [0,3,2]	10	5,2	164,5	7.200
rco_2_60		15,0	[16,0; 17,0]	56,5	[1,2,4], [0,5,1]	13	4,3	89,2	7.200
rco_2_70	2	17,5	[18,0; 18,0]	65,8	[0,4,3], [3,3,1]	14	4,7	68,1	7.200
rco_2_80		20,0	[21,0; 20,0]	69,8	[3,4,1], [1,4,3]	16	4,4	54,4	7.200
rco_2_90		22,5	[23,0; 24,0]	94,1	[5,3,1], [2,5,2]	18	5,2	47,8	7.200
rco_2_50		12,5	[13,0; 13,0; 14,0]	82,9	[0,3,2], [2,2,1], [0,2,4]	16	5,2	142,3	7.200
rco_2_60		15,0	[15,0; 15,0; 15,0]	92,6	[0,3,3], [2,2,2], [0,3,3]	18	5,1	81,0	7.200
rco_2_70	3	17,5	[18,0; 17,5; 17,5]	116,9	[2,3,2], [1,3,3], [3,2,2]	21	5,6	46,4	7.200
rco_2_80		20,0	[20,0; 20,0; 20,0]	137,6	[4,2,2], [1,3,4], [3,3,2]	24	5,7	0,0	233
rco_2_90		22,5	[22,5; 22,5; 22,5]	250,1	[2,3,3], [4,2,2], [2,3,3]	24	10,4	0,0	24
rco_3_50		12,5	[15,0]	24,4	[1,4,2]	7	3,5	0,0	871
rco_3_60		15,0	[15,0]	24,4	[1,4,2]	7	3,5	0,0	574
rco_3_70	1	17,5	[18,0]	26,4	[1,6,2]	9	2,9	0,0	1.934
rco_3_80		20,0	[21,0]	30,6	[1,6,3]	10	3,1	0,0	1.890
rco_3_90		22,5	[24,0]	32,6	[1,8,3]	12	2,7	0,0	3.078
rco_3_50		12,5	[13,0; 14,0]	49,0	[2,1,2], [0,3,3]	11	4,5	140,0	7.200
rco_3_60		15,0	[15,0; 16,0]	54,5	[1,4,2], [2,1,3]	13	4,2	100,0	7.200
rco_3_70	2	17,5	[20,0; 20,0]	68,6	[1,3,4], [0,5,4]	17	4,0	79,6	7.200
rco_3_80		20,0	[20,0; 20,0]	76,4	[1,5,3], [1,3,4]	17	4,5	58,0	7.201
rco_3_90		22,5	[22,5; 23,0]	85,1	[2,3,4], [1,5,4]	19	4,5	44,1	7.200
rco_3_50		12,5	[13,0; 13,0; 14,0]	96,1	[3,1,1], [3,3,0], [1,3,2]	17	5,7	146,9	7.200
rco_3_60		15,0	[16,0; 15,0; 15,0]	104,5	[3,3,1], [2,2,2], [2,2,2]	19	5,5	83,2	7.200
rco_3_70	3	17,5	[17,5; 17,5; 18,0]	120,1	[3,0,3], [1,3,3], [2,4,2]	21	5,7	47,9	7.200
rco_3_80		20,0	[20,0; 20,0; 20,0]	137,6	[4,2,2], [2,3,3], [2,3,3]	24	5,7	0,0	48
rco_3_90		22,5	[22,5; 22,5; 22,5]	250,1	[3,3,2], [3,2,3], [2,3,3]	24	10,4	0,0	13
rco_4_50		12,5	[14,0]	23,6	[3,2,1]	6	3,9	0,0	698
rco_4_60		15,0	[16,0]	26,8	[4,2,1]	7	3,8	0,0	1.831
rco_4_70	1	17,5	[20,0]	27,9	[2,5,1]	8	3,5	0,0	1.097
rco_4_80		20,0	[20,0]	27,9	[2,5,1]	8	3,5	0,0	716
rco_4_90		22,5	[24,0]	34,3	[4,5,1]	10	3,4	0,0	959
rco_4_50		12,5	[13,0; 16,0]	46,2	[1,3,1], [0,5,1]	11	4,2	150,4	7.204
rco_4_60		15,0	[16,0; 17,0]	48,4	[0,6,0], [2,2,3]	13	3,7	82,0	7.200
rco_4_70	2	17,5	[18,0; 18,0]	61,2	[5,2,1], [1,4,2]	15	4,1	58,0	7.200
rco_4_80		20,0	[21,0; 21,0]	72,0	[1,6,1], [4,2,3]	17	4,2	43,6	7.200
rco_4_90		22,5	[23,0; 23,0]	78,4	[2,6,1], [5,2,3]	19	4,1	35,8	7.200
rco_4_50		12,5	[13,0; 13,0; 13,0]	87,1	[1,2,2], [1,2,2], [1,3,1]	15	5,8	152,5	7.200
rco_4_60		15,0	[15,0; 15,0; 15,0]	90,2	[2,3,1], [2,2,2], [2,3,1]	18	5,0	94,1	7.200
rco_4_70	3	17,5	[18,0; 18,0; 18,0]	128,0	[2,2,3], [1,3,3], [2,3,2]	21	6,1	51,1	7.200
rco_4_80		20,0	[20,0; 20,0; 20,0]	137,6	[2,2,4], [4,2,2], [2,4,2]	24	5,7	0,0	172
rco_4_90		22,5	[22,5; 22,5; 22,5]	250,1	[1,3,4], [4,2,2], [3,3,2]	24	10,4	0,0	38
rco_5_50		12,5	[13,0]	25,2	[2,1,2]	5	5,0	0,0	935
rco_5_60		15,0	[17,0]	27,9	[1,2,4]	7	4,0	0,0	711
rco_5_70	1	17,5	[20,0]	31,3	[1,4,4]	9	3,5	0,0	1.518
rco_5_80		20,0	[20,0]	31,3	[1,4,4]	9	3,5	0,0	1.064
rco_5_90		22,5	[24,0]	36,1	[5,3,2]	10	3,6	0,0	2.145
rco_5_50		12,5	[13,0; 15,0]	51,9	[0,3,3], [1,4,2]	13	4,0	131,4	7.200
rco_5_60		15,0	[20,0; 19,0]	70,0	[5,2,1], [3,3,2]	16	4,4	101,1	7.200
rco_5_70	2	17,5	[18,0; 18,0]	75,6	[3,4,1], [1,4,3]	16	4,7	80,3	7.200
rco_5_80		20,0	[20,0; 20,0]	68,8	[5,2,1], [3,2,3]	16	4,3	42,4	7.200
rco_5_90		22,5	[25,0; 23,0]	89,4	[3,5,3], [5,2,2]	20	4,5	42,1	7.200
rco_5_50		12,5	[13,0; 14,0; 13,0]	86,5	[2,1,2], [1,3,2], [2,1,2]	16	5,4	142,2	7.200
rco_5_60		15,0	[15,0; 16,0; 17,0]	98,6	[3,2,1], [2,1,3], [3,0,3]	18	5,5	87,6	7.200
rco_5_70	3	17,5	[20,0; 18,0; 17,5]	107,1	[4,2,2], [3,4,1], [1,2,4]	23	4,7	41,8	7.200
rco_5_80		20,0	[20,0; 20,0; 20,0]	137,6	[3,3,2], [2,2,4], [3,3,2]	24	5,7	0,0	151
rco_5_90		22,5	[22,5; 22,5; 22,5]	250,1	[3,3,2], [2,2,4], [3,3,2]	24	10,4	0,0	12

Tabela 7 – Resultados obtidos variando o comprimento mínimo - Instância *blazewicz*.

Instância	Nb	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	GAP (%)	T (s)
$blazewicz_2$_50		12,5	[12,5]	40,2	[2,2,1]	5	8,0	42,0	7.200
$blazewicz_2$_60		15,0	[16,0]	37,9	[2,3,2]	7	5,4	0,0	6.634
$blazewicz_2$_70	1	17,5	[18,0]	40,4	[4,2,2]	8	5,1	0,0	5.071
$blazewicz_2$_80		20,0	[21,0]	58,1	[2,4,3]	9	6,5	49,0	7.200
$blazewicz_2$_90		22,5	[23,0]	60,6	[4,3,3]	10	6,1	35,0	7.200
$blazewicz_2$_50		12,5	[12,5; 13,0]	75,8	[2,2,1], [2,1,3]	11	6,9	133,0	7.200
$blazewicz_2$_60		15,0	[15,0; 15,0]	90,4	[2,3,1], [2,3,1]	12	7,5	99,0	7.200
$blazewicz_2$_70	2	17,5	[18,0; 17,5]	105,2	[4,3,0], [3,3,1]	14	7,5	76,0	7.200
$blazewicz_2$_80		20,0	[20,0; 20,0]	108,3	[3,2,4], [5,3,0]	17	6,4	57,0	7.200
$blazewicz_2$_90		22,5	[25,0; 22,5]	131,0	[3,4,4], [4,4,1]	20	6,6	46,0	7.200
$blazewicz_2$_50		12,5	[12,5; 12,5; 12,5]	120,6	[2,2,1], [2,2,1], [2,2,1]	15	8,0	107,0	7.200
$blazewicz_2$_60		15,0	[15,0; 15,0; 15,0]	138,1	[2,3,1], [3,2,1], [2,3,1]	18	7,7	67,0	7.200
$blazewicz_2$_70	3	17,5	[17,5; 18,0; 18,0]	146,4	[3,1,4], [3,4,0], [2,3,3]	23	6,4	24,0	7.200
$blazewicz_2$_80		20,0	[20,0; 20,0; 20,0]	221,6	[3,2,3], [2,3,3], [3,3,2]	24	9,2	0,0	63
$blazewicz_2$_90		22,5	[22,5; 22,5; 22,5]	334,1	[2,2,4], [5,3,0], [1,3,4]	24	13,9	0,0	10
$blazewicz_3$_50		12,5	[13,0]	36,4	[1,3,2]	6	6,1	39,0	7.200
$blazewicz_3$_60		15,0	[16,0]	39,9	[3,2,2]	7	5,7	30,0	7.200
$blazewicz_3$_70	1	17,5	[19,0]	45,8	[2,2,4]	8	5,7	40,0	7.200
$blazewicz_3$_80		20,0	[21,0]	49,6	[4,2,3]	9	5,5	38,0	7.200
$blazewicz_3$_90		22,5	[24,0]	55,6	[4,4,3]	11	5,1	26,0	7.200
$blazewicz_3$_50		12,5	[12,5; 13,0]	75,6	[3,1,1], [3,0,2]	10	7,6	142,0	7.200
$blazewicz_3$_60		15,0	[15,0; 15,0]	80,4	[3,3,1], [3,3,1]	14	5,7	103,0	7.200
$blazewicz_3$_70	2	17,5	[17,5; 17,5]	107,8	[2,4,2], [2,4,2]	16	6,7	78,0	7.200
$blazewicz_3$_80		20,0	[20,0; 20,0]	102,2	[4,3,2], [1,4,4]	18	5,7	52,0	7.200
$blazewicz_3$_90		22,5	[23,0; 23,0]	115,9	[4,3,3], [3,3,4]	20	5,8	39,0	7.200
$blazewicz_3$_50		12,5	[13,0; 13,0; 13,0]	117,7	[2,3,1], [1,1,3], [1,3,2]	17	6,9	107,0	7.200
$blazewicz_3$_60		15,0	[15,0; 15,0; 15,0]	131,5	[1,4,2], [3,1,2], [3,3,1]	20	6,6	64,0	7.200
$blazewicz_3$_70	3	17,5	[17,5; 17,5; 21,0]	161,6	[4,1,2], [2,2,3], [2,5,3]	24	6,7	31,0	7.200
$blazewicz_3$_80		20,0	[20,0; 20,0; 20,0]	221,6	[2,3,3], [2,3,3], [4,2,2]	24	9,2	0,0	36
$blazewicz_3$_90		22,5	[22,5; 22,5; 22,5]	334,1	[3,2,3], [2,3,3], [3,3,2]	24	13,9	0,0	13
$blazewicz_4$_50		12,5	[13,0]	28,1	[3,1,2]	6	4,7	46,0	7.200
$blazewicz_4$_60		15,0	[16,0]	40,9	[2,3,2]	7	5,8	40,0	7.200
$blazewicz_4$_70	1	17,5	[20,0]	46,3	[4,4,1]	9	5,1	66,0	7.200
$blazewicz_4$_80		20,0	[21,0]	39,2	[6,1,3]	10	3,9	17,0	7.200
$blazewicz_4$_90		22,5	[24,0]	52,0	[5,3,3]	11	4,7	34,0	7.200
$blazewicz_4$_50		12,5	[14,0; 12,5]	84,1	[2,2,2], [1,3,1]	11	7,6	145,0	7.200
$blazewicz_4$_60		15,0	[16,0; 15,0]	93,3	[3,3,1], [1,5,0]	13	7,2	111,0	7.200
$blazewicz_4$_70	2	17,5	[18,0; 19,0]	101,5	[3,1,4], [2,5,1]	16	6,3	82,0	7.200
$blazewicz_4$_80		20,0	[21,0; 21,0]	117,5	[3,4,2], [2,4,3]	18	6,5	63,0	7.200
$blazewicz_4$_90		22,5	[22,5; 23,0]	117,4	[4,3,3], [3,4,3]	20	5,9	42,0	7.200
$blazewicz_4$_50		12,5	[12,5; 14,0; 12,5]	127,8	[1,3,1], [2,2,2], [1,3,1]	16	8,0	108,0	7.200
$blazewicz_4$_60		15,0	[16,0; 16,0; 15,0]	137,2	[3,3,1], [2,2,3], [1,3,2]	20	6,9	68,0	7.200
$blazewicz_4$_70	3	17,5	[19,0; 19,0; 19,0]	176,6	[4,3,1], [2,3,3], [2,2,4]	24	7,4	37,0	7.200
$blazewicz_4$_80		20,0	[20,0; 20,0; 20,0]	221,6	[3,3,2], [3,2,3], [2,3,3]	24	9,2	0,0	29
$blazewicz_4$_90		22,5	[22,5; 22,5; 22,5]	334,1	[4,1,4], [2,3,2], [2,4,2]	24	13,9	0,0	10
$blazewicz_5$_50		12,5	[16,0]	36,5	[2,5,1]	8	4,6	28,0	7.200
$blazewicz_5$_60		15,0	[16,0]	42,6	[3,3,1]	7	6,1	36,0	7.200
$blazewicz_5$_70	1	17,5	[18,0]	44,1	[3,3,2]	8	5,5	45,0	7.200
$blazewicz_5$_80		20,0	[20,0]	46,3	[4,4,1]	9	5,1	25,0	7.200
$blazewicz_5$_90		22,5	[23,0]	55,8	[5,4,1]	10	5,6	32,0	7.200
$blazewicz_5$_50		12,5	[13,0; 13,0]	65,5	[1,2,3], [2,3,1]	12	5,5	151,0	7.200
$blazewicz_5$_60		15,0	[15,0; 15,0]	84,6	[2,4,1], [2,4,1]	14	6,0	114,0	7.200
$blazewicz_5$_70	2	17,5	[17,5; 17,5]	102,6	[2,4,2], [2,4,2]	16	6,4	80,0	7.200
$blazewicz_5$_80		20,0	[20,0; 20,0]	112,7	[3,4,2], [4,2,2]	17	6,6	59,0	7.200
$blazewicz_5$_90		22,5	[25,0; 22,5]	126,6	[4,4,3], [3,4,3]	21	6,0	45,0	7.200
$blazewicz_5$_50		12,5	[13,0; 13,0; 13,0]	105,4	[2,3,1], [3,1,1], [2,3,1]	17	6,2	110,0	7.200
$blazewicz_5$_60		15,0	[15,0; 15,0; 15,0]	130,7	[2,1,3], [3,1,2], [2,4,1]	19	6,9	63,0	7.200
$blazewicz_5$_70	3	17,5	[18,0; 17,5; 17,5]	165,9	[3,1,3], [2,4,2], [3,2,2]	22	7,5	34,0	7.200
$blazewicz_5$_80		20,0	[20,0; 20,0; 20,0]	221,6	[3,3,2], [2,4,3], [3,1,3]	24	9,2	0,0	36
$blazewicz_5$_90		22,5	[22,5; 22,5; 22,5]	334,1	[3,2,3], [2,2,4], [3,4,1]	24	13,9	0,0	11

Tabela 8 – Resultados obtidos variando o comprimento mínimo - Instância *marques*.

Instância	Min_c	z	Desperdício	Δ_w	Δ_{total}	DpP	GAP (%)	T (s)
$marques_2$_50	170,0	191,0	5913,2	4	4	1478,3	69,5	7.200
$marques_2$_60	204,0	204,0	7265,2	4	4	1816,3	54,7	7.200
$marques_2$_70	238,0	242,0	7729,5	5	5	1545,9	40,2	7.200
$marques_2$_80	272,0	290,0	9233,8	6	6	1573,6	36,3	7.200
$marques_2$_90	306,0	332,0	10114,1	7	7	1444,9	21,6	7.200

Como a instância $marques_2$ possui apenas um tipo de produto final, é possível gerar um plano simples (empacotar apenas uma unidade deste produto final) e repetir este padrão múltiplas vezes até preencher toda a mesa de corte. Entretanto, essa estratégia pode perder eficiência por não considerar encaixes que ocorrem apenas quando existem mais de uma unidade de produto final no plano de corte. Desta forma, foram gerados planos simples para comparar seus desperdícios por produto final com os planos apresentados na Tabela 8. Os planos simples foram gerados considerando um comprimento máximo de 61 e comprimento mínimo de 0. Foram utilizadas diferentes malhas regulares com discretizações gxg, sendo $g \in \{7;\ 3,5;\ 1,75\}$. Vale destacar que só foi possível utilizar discretizações mais refinadas (densas) devido à diminuição no comprimento do *bin* e à redução na demanda. Os planos são identificados por $marques_2$_0_g. As Figuras 23a, 23b e 23c ilustram os planos simples obtidos, enquanto a Tabela 9 mostra os resultados obtidos.

Tabela 9 – Resultados obtidos para a instância *marques* considerando diferentes malhas de pontos e um único produto final.

Instância	Min_c	z	Desperdício	Δ_w	Δ_{total}	DpP	GAP (%)	T (s)
$marques_2$_0_1,75	0,0	45,5	1244,3	1	1	1244,3	0,0	1.244
$marques_2$_0_3,5	0,0	50,5	1764,3	1	1	1764,3	0,0	55
$marques_2$_0_7,0	0,0	54,0	2128,3	1	1	2128,3	0,0	2

Os resultados permitem concluir que tratar o empacotamento de múltiplos produtos finais gera, na maioria das vezes, resultados melhores que repetir um plano simples para obter um plano de corte final. Isso é comprovado pois os planos $marques_2$_0_7,0 e $marques_2$_0_3,5 apresentam desperdícios por produto final maiores em comparação com os planos $marques_2$_50, $marques_2$_70, $marques_2$_80 e $marques_2$_90. Note que, se forem consideradas malhas com a mesma discretização, o plano simples $marques_2$_0_7,0 é o que apresentou menor custo benefício. Porém, uma vantagem de se utilizar o plano simples é a possibilidade de refinar a malha. Esse refinamento gera planos mais eficiente, como mostra o plano $marques_2$_0_1,75, que possui o melhor custo benefício entre todos. Entretanto, é válido ressaltar que esta abordagem só é possível devido ao fato da instância $marques_2$ possuir apenas um produto final.

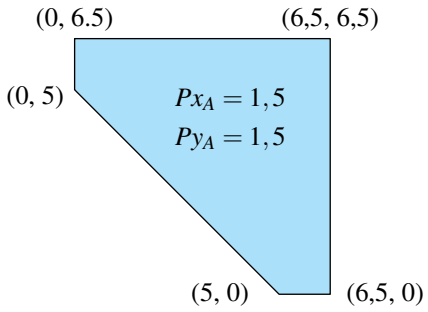
3.2.5 Fase IV - Malha alternativa

Como apontam Cherri *et al.* (2018), utilizar malhas regulares para todas as instâncias nem sempre é adequado. Isso se deve ao fato de que alguns itens não aproveitam bem este tipo

de malha e, desta forma, acabam com opções de alocação limitadas. Sendo assim, esta seção realiza testes utilizando a malha *piece based mesh* (PBM) proposta por Cherri *et al.* (2018).

Esta malha tem como objetivo gerar pontos baseados nas características dos itens. Logo, para cada tipo de item, é gerada uma malha diferente para evitar que posições pouco aproveitadas sejam atribuídas aos itens. Para isso, os espaçamentos entre os pontos das malhas são calculados utilizando a distância mínima entre os vértices de cada item. Sendo Px_i a menor distância positiva horizontal entre dois vértices do item de tipo i e Py_i a menor distância vertical, a malha $Malha_i$ para o item de tipo i é gerada utilizando os espaçamentos Px_i e Py_i. Seja H_i^{min} a coordenada y em que o IFP_i se inicia, H_i^{max} a coordenada y em que ele termina, L_i^{min} a coordenada x em que o IFP_i se inicia e L_i^{max} a coordenada x em que ele termina. Dado um item i, os pontos são gerados partindo da extremidade inferior esquerda do IFP_i (ou seja, $x = L_i^{min}$ e $y = H_i^{min}$). Dessa forma, partindo dessa coordenada, os pontos são gerados ao incrementar em Px_i a coordenada x e em Py_i a coordenada y até que se atinja os limites do IFP_i (no caso, L_i^{max} e H_i^{max}). Para criar uma malha que favoreça itens que não são simétricos, os pontos também são gerados partindo da extremidade superior esquerda (ou seja, $x = L_i^{min}$ e $y = H_i^{max}$) de forma análoga (nesse caso, não há um incremento de Py_i e sim um decremento de Py_i). Na Figura 24, é ilustrado o cálculo dos espaçamentos para um item do tipo A.

Figura 24 – Cálculo do espaçamento.



Vale ressaltar que Cherri *et al.* (2018) recalcula Px_i e Py_i no caso destes serem muito pequenos, pois assim é evitada a geração de malhas muito densas (com muitos pontos). Para a implementação neste trabalho, foi utilizado o seguinte cálculo: sendo lim_x e lim_y o limite de refinamento horizontal e vertical respectivamente, se $Px_i < lim_x$, então $\overline{Px_i} = \overline{n_x} * Px_i$, em que $\overline{n_x} = \min\{n \in \mathbb{N} : n * Px_i \geq lim_x\}$. Ou seja, no caso do espaçamento ser pequeno demais, é utilizado o múltiplo dele mais próximo do limite de espaçamento. De forma análoga, é calculado Py_i. Para este experimento, os limites de refinamento foram 7 para ambos os espaçamentos, pois este foi o valor utilizado para gerar a malha regular (valores menores gerariam malhas muito densas).

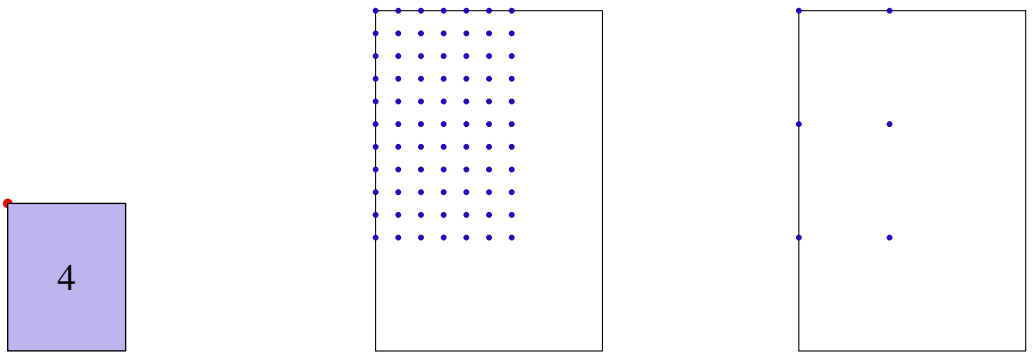
No Algoritmo 1, é descrito o código utilizado. Para cada item, é criada uma malha utilizando os espaçamentos Px_i e Py_i. Caso estes espaçamentos sejam muito pequenos, eles são recalculados para evitar a criação de malhas com muitos pontos. Além disso, note que os pontos

são criados começando tanto pela base ($y = H_i^{min}$) quanto pelo topo ($y = H_i^{max}$) do IFP_i. Um exemplo de malha regular e malha PBM geradas para o item 4 da instância *rco* é ilustrado na Figura 25.

Algoritmo 1 – Pseudo-código para gerar PBM

Dados de entrada: lim_x, lim_y, L_i^{min}, L_i^{max}, H_i^{min}, H_i^{max}, Px_i e Py_i, para cada item i.
`/* Conjunto de malhas. Guarda a malha de cada item. */`
Conjunto_de_malhas $\leftarrow \emptyset$
for *Cada item i* **do**
  $Malha_i \leftarrow \emptyset$
  if $Px_i < lim_x$ **then**
    $\overline{n_x} = \min\{n \in \mathbb{N} : n * Px_i \geq lim_x\}$
    $Px_i = \overline{n_x} * Px_i$
  end
  if $Py_i < lim_y$ **then**
    $\overline{n_y} = \min\{n \in \mathbb{N} : n * Py_i \geq lim_y\}$
    $Py_i = \overline{n_y} * Py_i$
  end
  $x \leftarrow l_i^{min}$
  while $x \leq l_i^{max}$ **do**
    $y \leftarrow h_i^{min}$
    while $y \leq h_i^{max}$ **do**
      $Malha_i \leftarrow (x, y)$
      $y \leftarrow y + Py_i$
    end
    $y \leftarrow h_i^{max}$
    while $y \geq h_i^{min}$ **do**
      $Malha_i \leftarrow (x, y)$
      $y \leftarrow y - Py_i$
    end
    $x \leftarrow x + Px_i$
  end
  Conjunto_de_malhas $\leftarrow Malha_i$
end
Devolve (Conjunto_de_malhas)

Figura 25 – Comparação entre as malhas.



(a) Item 4 da instância *rco*. (b) Malha regular para o item 4. (c) Malha PBM para o item 4.

Fonte: Elaborado pelo autor.

Para estudar a nova malha, foram utilizadas as instâncias da terceira fase com o tempo limite de 7200 segundos. Os resultados obtidos são resumidos nas Tabelas 10, 11 e 12.

Em relação às instâncias do tipo *rco* e *blazewicz*, o uso da PBM se mostrou ineficiente, pois as soluções que utilizaram a malha regular (Tabelas 6 e 7) se mostraram melhores tanto em relação ao desperdício quanto ao desperdício por produto final na maioria dos casos. Uma explicação para isso é o fato dessas instâncias se beneficiarem bastante da malha regular, uma vez que seus vértices são coordenadas inteiras e seus objetos de corte não possuem grandes dimensões. Desta forma, a malha regular permite encaixes mais eficientes.

Porém, para a instância do tipo $marques_2$, é observado que o uso da PBM é mais eficiente, pois tanto o desperdício quanto o desperdício por produto final foram melhorados ao utilizar a nova malha. Vale ressaltar que o plano de corte obtido para $marques_2$_90 possui um melhor desperdício por produto final que o plano simples obtido em $marques_2$_0_1,75.

Tabela 10 – Resultados computacionais utilizando a PBM - Instância *rco*.

Instância	Nb	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	GAP (%)	T (s)
rco_2_50		12,5	[13,0]	28,7	[2,2,1]	5	5,7	0,0	609
rco_2_60		15,0	[17,0]	29,1	[1,3,3]	7	4,2	0,0	705
rco_2_70	1	17,5	[21,0]	31,4	[1,5,2]	8	3,9	0,0	882
rco_2_80		20,0	[21,0]	31,4	[1,5,2]	8	3,9	0,0	180
rco_2_90		22,5	[23,0]	37,1	[1,5,3]	9	4,1	0,0	354
rco_2_50		12,5	[13,0; 13,0]	52,1	[2,2,1], [0,3,2]	10	5,2	67,6	7.200
rco_2_60		15,0	[15,0; 15,0]	65,4	[1,4,0], [2,2,2]	11	5,9	49,8	7.200
rco_2_70	2	17,5	[19,0; 18,0]	74,7	[4,3,0], [3,3,1]	14	5,3	46,6	7.200
rco_2_80		20,0	[20,0; 20,0]	84,4	[1,5,1], [5,2,1]	15	5,6	34,7	7.200
rco_2_90		22,5	[23,0; 24,0]	81,2	[1,5,3], [4,3,3]	19	4,3	23,8	7.200
rco_2_50		12,5	[13,0; 13,0; 14,0]	84,8	[0,3,2], [2,2,1], [1,3,1]	15	5,7	114,8	7.201
rco_2_60		15,0	[15,0; 15,0; 15,0]	99,8	[1,4,0], [2,2,2], [2,2,2]	17	5,9	67,4	7.204
rco_2_70	3	17,5	[18,0; 17,5; 18,0]	118,7	[2,3,2], [0,3,4], [5,2,0]	21	5,7	44,3	7.200
rco_2_80		20,0	[20,0; 20,0; 20,0]	137,6	[4,2,2], [2,3,3], [2,3,3]	24	5,7	0,0	35
rco_2_90		22,5	[22,5; 22,5; 22,5]	250,1	[4,2,2], [3,3,2], [1,3,4]	24	10,4	0,0	5
rco_3_50		12,5	[15,0]	24,4	[1,4,2]	7	3,5	0,0	426
rco_3_60		15,0	[15,0]	24,4	[1,4,2]	7	3,5	0,0	285
rco_3_70	1	17,5	[18,0]	26,4	[1,6,2]	9	2,9	0,0	344
rco_3_80		20,0	[24,0]	32,6	[1,8,3]	12	2,7	0,0	679
rco_3_90		22,5	[24,0]	32,6	[1,8,3]	12	2,7	0,0	235
rco_3_50		12,5	[14,0; 17,0]	56,0	[0,5,2], [1,3,3]	14	4,0	82,9	7.200
rco_3_60		15,0	[16,0; 17,0]	63,0	[2,3,2], [1,3,3]	14	4,5	61,4	7.200
rco_3_70	2	17,5	[20,0; 18,0]	67,9	[0,5,4], [2,2,3]	16	4,2	45,6	7.200
rco_3_80		20,0	[20,0; 20,0]	68,6	[1,3,4], [0,5,4]	17	4,0	16,8	7.200
rco_3_90		22,5	[24,0; 24,0]	89,6	[2,4,4], [2,4,4]	20	4,5	31,5	7.200
rco_3_50		12,5	[13,0; 13,0; 13,0]	90,7	[0,2,3], [3,1,1], [0,4,2]	16	5,7	119,5	7.200
rco_3_60		15,0	[16,0; 15,0; 16,0]	92,4	[2,1,3], [2,4,1], [2,1,3]	19	4,9	64,8	7.204
rco_3_70	3	17,5	[18,0; 19,0; 19,0]	110,6	[1,4,3], [3,1,3], [3,3,2]	23	4,8	36,7	7.200
rco_3_80		20,0	[20,0; 20,0; 20,0]	137,6	[1,3,4], [3,3,2], [4,2,2]	24	5,7	0,0	55
rco_3_90		22,5	[22,5; 22,5; 22,5]	250,1	[3,2,3], [2,2,3], [3,4,2]	24	10,4	0,0	8
rco_4_50		12,5	[13,0]	24,7	[1,3,1]	5	4,9	0,0	154
rco_4_60		15,0	[15,0]	27,9	[2,3,1]	6	4,6	0,0	193
rco_4_70	1	17,5	[18,0]	31,2	[1,4,2]	7	4,5	0,0	278
rco_4_80		20,0	[21,0]	32,2	[1,6,1]	8	4,0	0,0	280
rco_4_90		22,5	[23,0]	35,4	[2,6,1]	9	3,9	0,0	182
rco_4_50		12,5	[16,0; 13,0]	46,2	[0,5,1], [1,3,1]	11	4,2	61,4	7.200
rco_4_60		15,0	[16,0; 16,0]	48,3	[4,1,2], [0,6,0]	13	3,7	0,0	4.486
rco_4_70	2	17,5	[18,0; 20,0]	65,6	[1,4,2], [2,4,2]	15	4,4	25,7	7.200
rco_4_80		20,0	[20,0; 20,0]	68,8	[2,4,2], [2,4,2]	16	4,3	9,6	7.200
rco_4_90		22,5	[23,0; 24,0]	80,8	[1,5,3], [3,3,4]	19	4,3	23,1	7.200
rco_4_50		12,5	[13,0; 16,0; 13,0]	90,4	[1,2,2], [0,4,2], [1,2,2]	16	5,7	113,9	7.205
rco_4_60		15,0	[15,0; 15,0; 16,0]	96,8	[2,2,2], [2,2,2], [0,4,2]	18	5,4	59,6	7.200
rco_4_70	3	17,5	[17,5; 19,0; 18,0]	112,9	[2,2,3], [4,2,2], [1,4,2]	22	5,1	38,0	7.200
rco_4_80		20,0	[20,0; 20,0; 20,0]	137,6	[3,2,3], [2,3,3], [3,3,2]	24	5,7	0,0	28
rco_4_90		22,5	[22,5; 22,5; 22,5]	250,1	[4,1,3], [2,4,2], [2,3,3]	24	10,4	0,0	6
rco_5_50		12,5	[13,0]	25,2	[2,1,2]	5	5,0	0,0	178
rco_5_60		15,0	[16,0]	28,1	[3,3,1]	7	4,0	0,0	161
rco_5_70	1	17,5	[20,0]	33,9	[5,2,1]	8	4,2	0,0	316
rco_5_80		20,0	[20,0]	33,9	[5,2,1]	8	4,2	0,0	228
rco_5_90		22,5	[24,0]	39,7	[7,1,1]	9	4,4	0,0	296
rco_5_50		12,5	[13,0; 13,0]	50,4	[2,1,2], [2,1,2]	10	5,0	67,9	7.200
rco_5_60		15,0	[16,0; 16,0]	56,7	[2,3,2], [3,3,1]	14	4,0	39,2	7.200
rco_5_70	2	17,5	[20,0; 18,0]	71,2	[4,2,2], [4,4,0]	16	4,4	44,1	7.200
rco_5_80		20,0	[20,0; 20,0]	73,4	[5,2,1], [3,0,4]	15	4,9	19,3	7.200
rco_5_90		22,5	[23,0; 23,0]	84,8	[3,2,4], [5,2,2]	18	4,7	23,3	7.200
rco_5_50		12,5	[14,0; 12,5; 14,0]	104,1	[3,3,0], [2,2,1], [3,3,0]	17	6,1	106,9	7.200
rco_5_60		15,0	[15,0; 17,0; 16,0]	94,0	[2,2,2], [3,0,3], [3,3,1]	19	4,9	65,4	7.200
rco_5_70	3	17,5	[18,0; 18,0; 18,0]	121,6	[1,2,4], [3,4,1], [4,2,1]	22	5,5	41,0	7.200
rco_5_80		20,0	[20,0; 20,0; 20,0]	137,6	[2,3,3], [1,5,3], [5,0,2]	24	5,7	0,0	160
rco_5_90		22,5	[22,5; 22,5; 22,5]	250,1	[4,3,1], [2,2,4], [2,3,3]	24	10,4	0,0	7

Tabela 11 – Resultados computacionais utilizando a PBM - Instância *blazewicz*.

Instância	Nb	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	GAP (%)	T (s)
$blazewicz_2$_50		12,5	[14,0]	32,7	[3,2,1]	6	5,5	0,0	955
$blazewicz_2$_60		15,0	[18,0]	40,4	[4,2,2]	8	5,1	0,0	2.545
$blazewicz_2$_70	1	17,5	[18,0]	40,4	[4,2,2]	8	5,1	0,0	1.472
$blazewicz_2$_80		20,0	[20,0]	48,1	[4,2,3]	9	5,3	0,0	3.447
$blazewicz_2$_90		22,5	[23,0]	55,4	[6,2,2]	10	5,5	13,9	7.200
$blazewicz_2$_50		12,5	[12,5; 13,0]	77,7	[2,2,1], [2,3,0]	10	7,8	98,9	7.200
$blazewicz_2$_60		15,0	[17,0; 15,0]	90,4	[3,3,1], [2,3,1]	13	7,0	73,0	7.200
$blazewicz_2$_70	2	17,5	[18,0; 18,0]	90,4	[4,2,2], [3,4,0]	15	6,0	43,8	7.200
$blazewicz_2$_80		20,0	[20,0; 20,0]	110,2	[5,3,0], [3,4,1]	16	6,9	38,5	7.200
$blazewicz_2$_90		22,5	[23,0; 23,0]	145,4	[4,4,1], [4,4,1]	18	8,1	41,0	7.200
$blazewicz_2$_50		12,5	[12,5; 12,5; 12,5]	120,6	[2,2,1], [2,2,1], [2,2,1]	15	8,0	93,3	7.200
$blazewicz_2$_60		15,0	[15,0; 15,0; 15,0]	138,1	[2,3,1], [3,2,1], [2,3,1]	18	7,7	56,2	7.200
$blazewicz_2$_70	3	17,5	[18,0; 17,5; 17,5]	161,2	[3,2,3], [4,2,1], [1,4,2]	22	7,3	24,3	7.200
$blazewicz_2$_80		20,0	[20,0; 20,0; 20,0]	221,6	[4,2,2], [1,3,4], [3,3,2]	24	9,2	0,0	53
$blazewicz_2$_90		22,5	[22,5; 22,5; 22,5]	334,1	[4,3,1], [2,3,3], [2,2,4]	24	13,9	0,0	21
$blazewicz_3$_50		12,5	[14,0]	36,2	[2,4,1]	7	5,2	0,0	1.595
$blazewicz_3$_60		15,0	[15,0]	40,2	[3,3,1]	7	5,7	0,0	3.563
$blazewicz_3$_70	1	17,5	[18,0]	46,1	[2,3,3]	8	5,8	0,0	4.965
$blazewicz_3$_80		20,0	[21,0]	52,1	[2,5,3]	10	5,2	0,0	5.289
$blazewicz_3$_90		22,5	[23,0]	60,1	[4,3,3]	10	6,0	16,2	7.200
$blazewicz_3$_50		12,5	[13,0; 13,0]	81,2	[1,1,3], [1,1,3]	10	8,1	94,4	7.200
$blazewicz_3$_60		15,0	[15,0; 16,0]	82,6	[3,3,1], [1,5,2]	15	5,5	66,9	7.200
$blazewicz_3$_70	2	17,5	[17,5; 17,5]	107,8	[2,4,2], [2,4,2]	16	6,7	53,2	7.200
$blazewicz_3$_80		20,0	[20,0; 20,0]	106,4	[4,1,3], [1,4,4]	17	6,3	29,0	7.200
$blazewicz_3$_90		22,5	[22,5; 23,0]	138,9	[3,2,4], [3,4,3]	19	7,3	35,4	7.200
$blazewicz_3$_50		12,5	[13,0; 14,0; 13,0]	119,3	[2,3,1], [2,2,2], [4,0,1]	17	7,0	94,9	7.200
$blazewicz_3$_60		15,0	[15,0; 15,0; 15,0]	131,5	[3,3,1], [3,1,2], [1,4,2]	20	6,6	48,5	7.200
$blazewicz_3$_70	3	17,5	[18,0; 18,0; 18,0]	151,1	[2,3,3], [3,1,3], [3,3,2]	23	6,6	17,9	7.200
$blazewicz_3$_80		20,0	[20,0; 20,0; 20,0]	221,6	[4,2,2], [2,3,3], [2,3,3]	24	9,2	0,0	19
$blazewicz_3$_90		22,5	[22,5; 22,5; 22,5]	334,1	[2,4,3], [3,3,2], [3,1,3]	24	13,9	0,0	6
$blazewicz_4$_50		12,5	[13,0]	51,2	[1,3,1]	5	10,2	141,9	18
$blazewicz_4$_60		15,0	[13,0]	51,2	[1,3,1]	5	10,2	141,9	18
$blazewicz_4$_70	1	17,5	[18,0]	44,1	[3,3,2]	8	5,5	0,0	2.116
$blazewicz_4$_80		20,0	[20,0]	47,3	[4,3,2]	9	5,3	0,0	4.379
$blazewicz_4$_90		22,5	[24,0]	53,7	[6,3,2]	11	4,9	1,3	7.200
$blazewicz_4$_50		12,5	[12,5; 14,0]	85,1	[1,3,1], [2,1,3]	11	7,7	108,0	7.200
$blazewicz_4$_60		15,0	[15,0; 15,0]	103,4	[1,4,1], [1,4,1]	12	8,6	84,2	7.200
$blazewicz_4$_70	2	17,5	[19,0; 17,5]	122,5	[3,2,3], [2,4,1]	15	8,2	62,3	7.200
$blazewicz_4$_80		20,0	[21,0; 21,0]	121,9	[4,3,2], [3,4,2]	18	6,8	47,3	7.200
$blazewicz_4$_90		22,5	[23,0; 23,0]	125,9	[3,3,4], [4,3,3]	20	6,3	31,4	7.200
$blazewicz_4$_50		12,5	[12,5; 16,0; 12,5]	134,4	[2,3,0], [3,2,2], [2,3,0]	17	7,9	95,7	7.200
$blazewicz_4$_60		15,0	[16,0; 15,0; 15,0]	149,0	[3,3,1], [1,3,2], [1,2,3]	19	7,8	62,1	7.200
$blazewicz_4$_70	3	17,5	[17,5; 17,5; 17,5]	189,5	[2,3,2], [1,2,4], [2,3,2]	21	9,0	38,3	7.200
$blazewicz_4$_80		20,0	[20,0; 20,0; 20,0]	221,6	[3,1,4], [2,5,1], [3,2,3]	24	9,2	0,0	38
$blazewicz_4$_90		22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [3,4,1], [3,1,4]	24	13,9	0,0	14
$blazewicz_5$_50		12,5	[14,0]	33,4	[3,2,1]	6	5,6	0,0	850
$blazewicz_5$_60		15,0	[15,0]	40,7	[3,1,2]	6	6,8	0,0	1.348
$blazewicz_5$_70	1	17,5	[18,0]	43,2	[5,1,1]	7	6,2	0,0	1.975
$blazewicz_5$_80		20,0	[20,0]	46,3	[4,4,1]	9	5,1	0,0	1.125
$blazewicz_5$_90		22,5	[23,0]	55,1	[4,3,3]	10	5,5	0,0	4.089
$blazewicz_5$_50		12,5	[13,0; 14,0]	72,6	[3,1,1], [3,2,1]	11	6,6	110,4	7.200
$blazewicz_5$_60		15,0	[15,0; 16,0]	88,7	[2,1,3], [4,0,2]	12	7,4	60,9	7.200
$blazewicz_5$_70	2	17,5	[18,0; 18,0]	88,9	[3,3,2], [4,4,0]	16	5,6	46,5	7.200
$blazewicz_5$_80		20,0	[20,0; 21,0]	113,0	[5,2,1], [3,3,3]	17	6,6	42,4	7.200
$blazewicz_5$_90		22,5	[23,0; 24,0]	112,1	[4,3,3], [4,5,2]	21	5,3	21,9	7.200
$blazewicz_5$_50		12,5	[13,0; 12,5; 13,0]	124,8	[2,1,2], [3,2,0], [3,1,1]	15	8,3	94,4	7.200
$blazewicz_5$_60		15,0	[16,0; 15,0; 15,0]	138,7	[3,3,1], [2,1,3], [3,2,1]	19	7,3	55,8	7.200
$blazewicz_5$_70	3	17,5	[17,5; 18,0; 18,0]	159,6	[3,2,2], [2,3,3], [2,3,3]	23	6,9	22,9	7.200
$blazewicz_5$_80		20,0	[20,0; 20,0; 20,0]	221,6	[2,4,2], [2,2,4], [4,2,2]	24	9,2	0,0	23
$blazewicz_5$_90		22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [3,3,2], [3,2,3]	24	13,9	0,0	8

Tabela 12 – Resultados computacionais utilizando a PBM - Instância *marques*.

Instância	Min_c	z	Desperdício	Δ_w	Δ_{total}	DpP	GAP (%)	T (s)
$marques_2$_50	170,0	[183,0]	5081,2	[4]	4	1270,3	131,6	7.200
$marques_2$_60	204,0	[204,0]	7265,2	[4]	4	1816,3	99,5	7.200
$marques_2$_70	238,0	[238,0]	7313,5	[5]	5	1462,7	78,0	7.200
$marques_2$_80	272,0	[273,0]	7465,8	[6]	6	1244,3	59,0	7.200
$marques_2$_90	306,0	[313,0]	8138,1	[7]	7	1162,6	44,4	7.200

3.3 Considerações

Neste capítulo, foi proposto um modelo matemático para o problema utilizando como base o modelo proposto por Toledo *et al.* (2013). As adaptações feitas tem como objetivo impor que os itens que compõem o mesmo produto final sejam empacotados em um mesmo *bin*. Os resultados computacionais mostram que caso não seja considerado o corte de produtos finais (ou seja, o empacotamento em conjunto), raramente são obtidos todos os itens de um produto em um mesmo plano de corte. Portanto, considerar o empacotamento de produtos finais mostra-se relevante para garantir a qualidade dos produtos finais e também para manter a fluidez na linha de produção, uma vez que assim é garantido que a etapa de costura seja iniciada logo após o corte dos itens.

Como esperado, foi observado que resolver o problema considerando múltiplos objetos é computacionalmente mais difícil devido à dimensão elevada do problema resultante que resolver com apenas um objeto, com exceção do caso em que o espaço disponível nos objetos é suficientemente grande para empacotar a demanda máxima de todos os produtos com folga.

Os experimentos computacionais também mostram que o comprimento mínimo do objeto deve ser definido com cuidado, pois valores baixos apresentam menor desperdício de matéria-prima, porém trazem, em geral, um menor custo benefício em relação ao desperdício associado a cada produto cortado. Já valores maiores levam a soluções com maior desperdício, porém com menores desperdícios por produto.

Como discutido na literatura, a escolha da malha de pontos se mostrou relevante para o problema estudado. Os testes computacionais mostraram que utilizar a malha PBM para a instância $marques_2$ é vantajoso, entretanto o mesmo não ocorre para as instância do tipo *rco* e *blazewicz*. Dessa forma, assim como Cherri *et al.* (2018) citam, nem sempre uma mesma malha é adequada para todas as instâncias.

Um ponto importante a ser observado é que não foi possível obter uma solução ótima para a maioria das instâncias. Portanto, uma abordagem heurística se mostra mais promissora ao tratar instâncias de dimensões maiores. No próximo capítulo, é apresentado um método heurístico para o problema.

CAPÍTULO

4

MÉTODO HEURÍSTICO

Como visto no capítulo anterior, resolver o problema de empacotamento de produtos finais com múltiplos *bins* utilizando um modelo matemático se mostrou desafiador, uma vez que não foi possível obter uma solução ótima para a maioria das instâncias-teste. Desta forma, neste capítulo, é proposta uma heurística *biased random key genetic algorithm* (BRKGA) para o problema com o objetivo de obter soluções de alta qualidade e com um baixo tempo de execução. O BRKGA é bastante utilizado na literatura para tratar problemas de empacotamento com itens irregulares (por exemplo, Mundim, Andretta e Queiroz (2017), Souza Queiroz e Andretta (2020) e Lu, Hu e Ng (2023)) devido à sua eficiência e facilidade de implementação e, portanto, é o escopo de estudo neste trabalho.

4.1 *Biased random key genetic algorithm*

A heurística *biased random key genetic algorithm* (BRKGA) foi proposta por Gonçalves e Resende (2011) e consiste em representar as soluções de um problema de otimização por meio de chaves aleatórias. Cada conjunto de chaves é chamado de cromossomo e cada chave aleatória do cromossomo é chamada de alelo. Cada cromossomo possui um valor de avaliação (*fitness*), sendo que menores (ou maiores) valores representam melhores soluções.

O conceito da heurística se baseia em evoluir o conjunto de soluções ao longo de gerações. Para isso, em cada geração, uma parte da população de cromossomos é denominada elite (que possuem os melhores valores de *fitness*), enquanto o restante é denominada não-elite; assim, elementos elites são cruzados (*crossover*) com não-elite para que seus descendentes herdem com a maior probabilidade as qualidades dos elites e tenham chances de obter boas características dos não-elite. Vale ressaltar que a probabilidade dos filhos obterem características do pai elite é maior. Além disso, em cada geração, são adicionados elementos mutantes (novos cromossomos com alelos gerados aleatoriamente). Assim, a nova população é formada pelos elementos elite, elementos de *crossover* e elementos mutantes (vale ressaltar que a quantidade

de cromossomos na população é fixa e igual em todas as gerações). Este procedimento é feito até que o número máximo de gerações seja atingido ou outro critério de parada seja ativado (por exemplo, tempo de execução ou número de gerações sem melhora na solução). O Algoritmo 2 mostra o pseudo-código do BRKGA.

Para realizar o *crossover*, é selecionado um indivíduo da população elite e outro da população não-elite aleatoriamente; desta forma, os alelos do cromossomo filho são escolhidos com probabilidade de *prob* de ser do indivíduo elite e $1 - prob$ de ser do não-elite. O Algoritmo 3 mostra o pseudo-código da operação de *crossover*. Já a criação de elementos mutantes é feita gerando cromossomos com valores aleatórios; a população inicial é gerada da mesma forma.

Algoritmo 2 – BRKGA

```
/* Tam_pop é o tamanho da população. Geracoes é a quantidade de gerações. prob é a
   probabilidade de herança do pai elite. pe é a porcentagem da população que se
   tornarão elementos elite. pm é a porcentagem da população que será gerada como
   elementos mutantes.                                                          */
Parâmetros: Tam_pop, Geracoes, prob, pe, pm
pop ← inicializar_pop /* População gerada aleatoriamente                       */
Decodifica (pop) /* Cálculo do fitness                                         */
i ← 0 /* Contador de gerações                                                  */
melhor_solucao ← ∅
while i < Geracoes (ou até outro critério de parada ser atingido) do
    Ordena pop pelo fitness
    best_pop ← melhor solução da população
    if best_pop_fitness é melhor que melhor_solucao_fitness then
        melhor_solucao ← best_pop
    end
    pop_elite ← pe dos melhores cromossomos de pop
    pop_nao_elite ← cromossomos não-elite de pop
    pop_mutante ← Gera pm de Tam_pop como cromossomos mutantes
    Decodifica pop_mutante
    nova_pop ← pop_elite + pop_mutante
    pop_crossover ← Gera a quantidade faltante de elementos da nova_pop utilizando crossover /* Realiza o
       crossover com probabilidade prob                                         */
    Decodifica (pop_crossover)
    pop ← nova_pop + pop_crossover
    i ← i + 1
end
Devolve (melhor_solucao)
```

Algoritmo 3 – Crossover

```
Dados de entrada: Cromossomo elite (ce) e cromossomo não-elite (cn).
Parâmetros: Probabilidade do cromossomo filho herdar alelo do pai elite (prob)
for Cada alelo j no cromossomo do
    if rand() < prob then
        /* Cromossomo filho herda j-ésimo alelo do pai elite                   */
        cromossomo_filho[j] ← ce[j]
    else
        /* Cromossomo filho herda j-ésimo alelo do pai não-elite               */
        cromossomo_filho[j] ← cn[j]
    end
end
Devolve (cromossomo_filho)
```

4.1.1 Representação da solução pelo cromossomo

Para representar as soluções do problema estudado por meio de chaves aleatórias, os cromossomos foram divididos em duas partes. Na primeira parte, são representados os produtos finais, enquanto a segunda parte representa os itens que compõem cada um dos produtos. Cada alelo da primeira parte representa um produto final, então são necessários $\sum_{w \in W} d_w^{max}$ alelos, ou seja, a soma das demandas máximas dos produtos finais. Na Figura 26, é ilustrado um exemplo em que a demanda máxima do Produto 1 é um, do Produto 2 é três e do Produto 3 é dois. Portanto, a primeira parte do cromossomo tem seis alelos. O número abaixo do respectivo produto indica o valor do alelo.

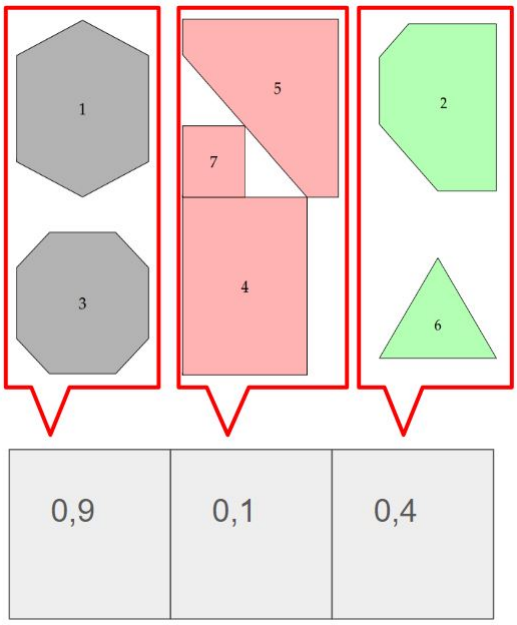
Figura 26 – Exemplo de cromossomo (primeira parte).

Fonte: Elaborada pelo autor.

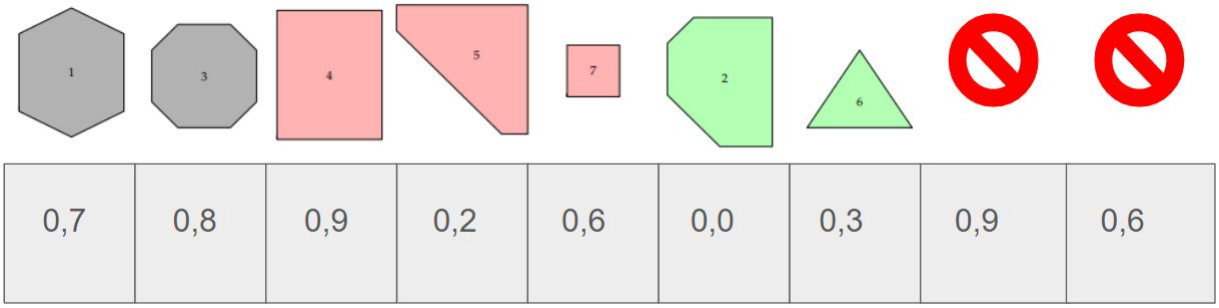
A segunda parte do cromossomo é utilizada para representar a ordem em que os itens serão alocados no *bin* seguindo a regra *bottom-left*. Assim como proposto por Souza Queiroz e Andretta (2020), esta parte do cromossomo será utilizada para determinar a ordem de inserção dos itens utilizando a heurística *bottom-left* e também a quantidade de posições ignoradas para cada item. De forma resumida, Souza Queiroz e Andretta (2020) permitem que cada item tenha a possibilidade de ignorar uma quantidade de posições no momento em que são alocados pela regra *bottom-left*. Por exemplo, se o item do tipo 1 ignora uma posição, então ele será alocado na segunda posição mais abaixo e à esquerda do objeto de corte. A quantidade de alelos desta parte é dada por $N_i + s$, sendo que N_i é a quantidade total de itens (ou seja, considerando a demanda máxima de produtos finais) e s é quantidade total de posições ignoradas. A quantidade total de itens pode ser calculada por $N_i = \sum_{w \in W} \sum_{i \in \mathscr{I}} n_{wi} d_w^{max}$, enquanto que a quantidade de posições ignoradas é dada por $s = \lfloor \beta * N_i \rfloor$ (fórmula utilizada por Souza Queiroz e Andretta (2020)), em que $\beta \in \mathbb{R}_+$ é um parâmetro a ser definido. Note que o valor de β é responsável por determinar a proporção entre a quantidade de posições ignoradas no total e a quantidade de itens a serem empacotados. Entretanto, Souza Queiroz e Andretta (2020) estudaram o caso para um único *bin*. Posteriormente, será explicado como a técnica foi adaptada para tratar múltiplos *bins*.

Na Figura 27, é ilustrado um exemplo de cromossomo considerando três produtos finais diferentes (sendo a demanda máxima de cada um deles de uma unidade) e $\beta = 0,3$. Como existem sete itens no total (dois no primeiro e no terceiro produto e três no segundo produto), são necessários sete alelos para representá-los. Em relação à quantidade de posições ignoradas (identificadas na figura pelo símbolo de proibido), para este caso, o total é igual a $2 = \lfloor \beta * N_i \rfloor = \lfloor 0,3 * 7 \rfloor$ alelos.

Figura 27 – Exemplo de cromossomo.



(a) Primeira parte.



(b) Segunda parte.

Fonte: Elaborada pelo autor.

4.1.2 Decodificação

Para decodificar a primeira parte do cromossomo, os alelos são ordenados de forma crescente segundo o valor de suas chaves. Desta forma, a ordem em que os produtos aparecem será a ordem de empacotamento. Este procedimento é ilustrado na Figura 26, em que a ordem de prioridade dos produtos é 2, 3, 2, 3, 2 e 1. Assim, o algoritmo tentará empacotar todos eles em um

bin vazio. Entretanto, caso não seja possível, o algoritmo tentará empacotar os produtos 2, 3, 2, 3 e 2 em um *bin*, e o produto 1 em outro, pois este possui a menor prioridade (ou seja, é removido o produto de menor prioridade sempre que o empacotamento for mal sucedido). Isto é feito até que seja possível empacotar todos os produtos designados. Caso os produtos empacotados atendam todas as demandas mínimas, então a decodificação é encerrada e o *fitness* da solução é calculado. Caso contrário, são gerados mais planos de corte considerando o restante dos produtos finais até que as demandas mínimas sejam atendidas. Note que não existe a possibilidade de ocorrer infactibilidade da solução uma vez que não é possível empacotar mais produtos que a demanda máxima e todas as demandas mínimas serão satisfeitas.

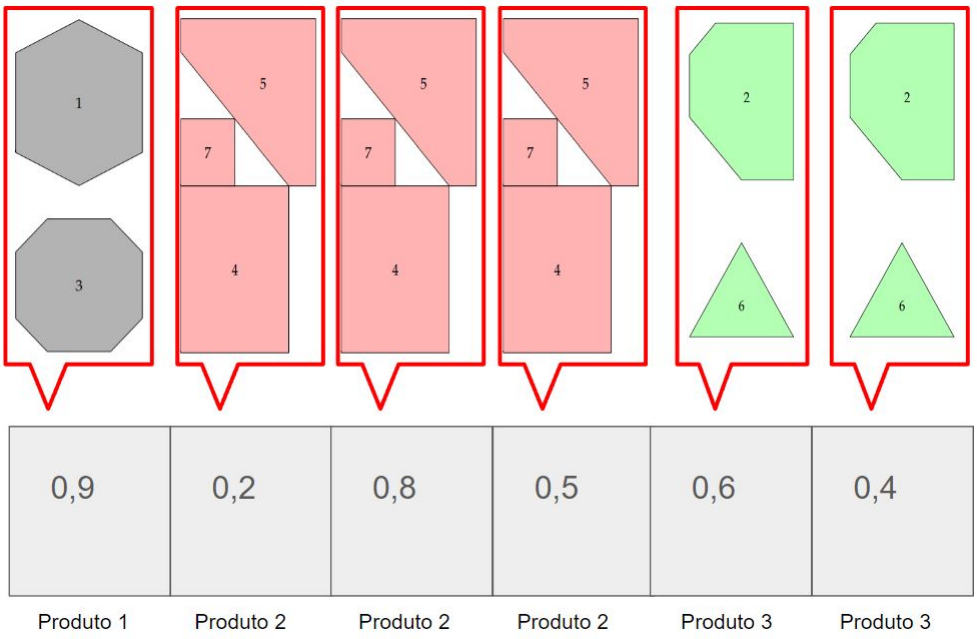
Por exemplo, considerando o conjunto de produtos finais da instância rco_2 (Tabela 3) e supondo uma demanda máxima de um para o produto 1, três para o produto 2 e dois para o produto 3, um exemplo de cromossomo para representar isso é ilustrado na Figura 28a. Para obter a ordem de prioridade, é necessário ordenar os alelos de forma crescente. Isto é feito na Figura 28b, obtendo a ordem de prioridade 2, 3, 2, 3, 2 e 1.

A designação dos produtos finais a cada objeto de corte é feita por meio da estimativa do comprimento necessário para empacotar cada produto individualmente. Com a finalidade de calcular o comprimento necessário do objeto para empacotar cada um dos produtos finais individualmente, foi solucionado o modelo matemático (Seção 3.1) considerando $Min_c = 0$. Desta forma, os produtos são adicionados ao objeto até que a soma do comprimento utilizado seja igual ou superior ao comprimento mínimo permitido ($p * Max_c$). Assim, é gerada uma lista de produtos finais a serem empacotados no *bin*. Na Figura 29, é ilustrado um exemplo de estimativa feita para estes produtos considerando os produtos finais da instância rco_2.

Entretanto, é provável que o comprimento real necessário para empacotar os produtos em conjunto seja menor que a estimativa, pois é possível realizar mais encaixes entre os itens ao considerar o empacotamento de múltiplos produtos finais simultaneamente. Caso isso ocorra, haverá uma perda no aproveitamento do objeto de corte, uma vez que é utilizado uma menor parte do objeto. Portanto, para gerar planos de corte mais eficientes, foi implementada uma tabela *hash* para mapear quanto de comprimento um conjunto de produtos finais ocupa. Ou seja, a tabela utiliza como chave a quantidade de cada produto final empacotado e tem como valor o comprimento utilizado para empacotá-los. Considerando uma instância com 3 produtos finais diferentes, a chave é dada por um vetor de tamanho 3 cujas entradas representam a quantidade de seu respectivo produto empacotado; já o valor é o comprimento necessário para empacotar os produtos da chave. Esta tabela é construída à medida que o problema é resolvido. Mais especificamente, toda vez que um empacotamento dos produtos em um *bin* for bem sucedido, é criado um novo elemento na tabela para mapear este plano de corte. Caso já exista uma chave que representa os produtos empacotados, o valor do elemento é atualizado realizando uma média ponderada entre o novo valor e o antigo.

Na Figura 30, é ilustrado um exemplo de tabela *hash* do problema. Note que, ao empaco-

Figura 28 – Exemplo de cromossomo (primeira parte).



(a) Cromossomo pré-ordenação.

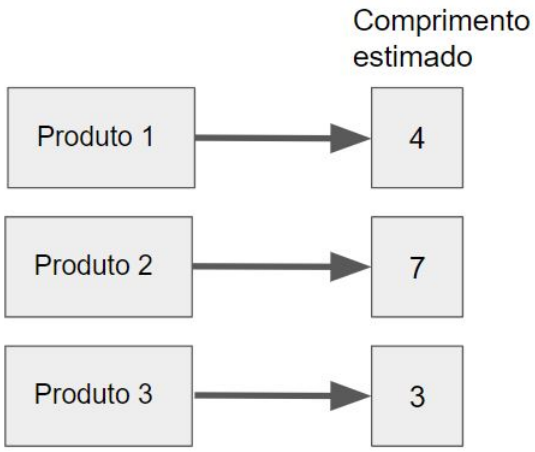
(b) Cromossomo pós-ordenação.

Fonte: Elaborada pelo autor.

tar um produto do tipo 1 e outro do tipo 3, seria necessário um comprimento de 7 unidades se fosse considerada a estimativa apresentada na Figura 29, entretanto foi guardada a informação na tabela *hash* que este empacotamento pode ser feito utilizando um comprimento de 4 unidades (primeira chave da Figura 30).

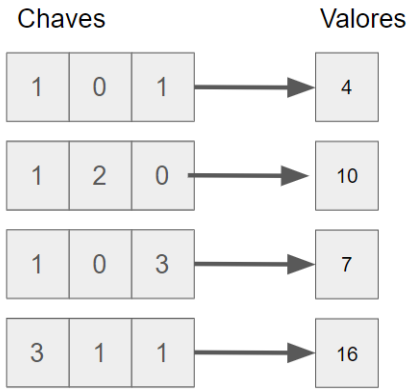
A tabela *hash* é utilizada toda vez que a estimativa de comprimento necessário atinge ou

Figura 29 – Exemplo de estimativa para empacotar cada produto individualmente.



Fonte: Elaborada pelo autor.

Figura 30 – Exemplo de tabela *hash*.



Fonte: Elaborada pelo autor.

ultrapassa o comprimento mínimo, pois quando isso ocorre, o valor da estimativa é atualizado com o valor guardado na tabela. Assim, se este valor é menor, o algoritmo continua a inserir itens no objeto de corte até que a estimativa de comprimento atinja o mínimo permitido novamente. Note que os valores da tabela precisam ser atualizados ao longo das gerações. Esta atualização é dada por:

$$HASH[key] = \alpha * novo\_comprimento + (1-\alpha) * antigo\_comprimento,\ \alpha \in [0,1].$$

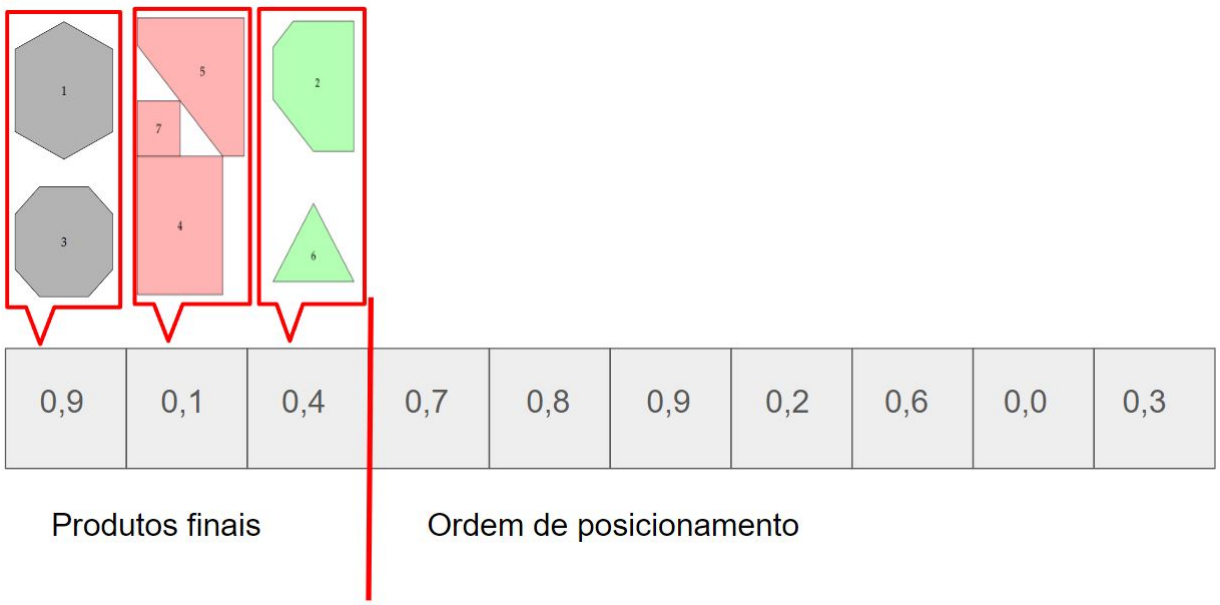
Desta forma, busca-se ponderar o histórico do comprimento utilizado com o novo comprimento, pois é preciso lembrar que o comprimento final é dado pelo empacotamento das peças que depende da ordem em que elas são empacotadas.

A segunda parte do cromossomo é utilizada para determinar a posição dos itens dentro do objeto de corte. Assim como nos trabalhos da literatura (Mundim, Andretta e Queiroz (2017),

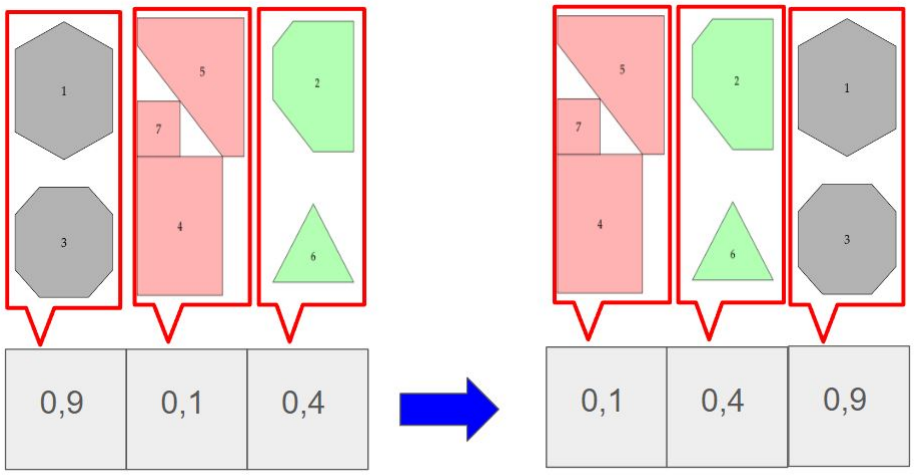
Souza Queiroz e Andretta (2020) e Lu, Hu e Ng (2023)), os alelos são utilizados para determinar a ordem de alocação seguida pela heurística *bottom-left*. Porém a rotação é fixa por conta da aplicação que é estudada neste trabalho. Note que o número de alelos dessa parte deve ser igual à quantidade total de itens necessários para gerar as demandas máximas dos produtos finais ($\sum_{w \in W} \sum_{i \in \mathscr{I}} n_{wi} d_w^{max}$). Assim, dada a lista de produtos finais que devem ser empacotados no *bin* (obtida na primeira parte do cromossomo), esta é convertida em uma lista de itens para então serem posicionados com a heurística *bottom-left*. A ordem dos itens é determinada pela ordem que os produtos finais se encontram; por exemplo, se os produtos finais aparecem na ordem 1 e 2, sendo que o produto 1 é composto pelos itens 1 e 3, enquanto o produto 2 é composto pelos itens 2 e 4, então a lista de itens será dada por 1, 3, 2 e 4; note que a ordem dos itens dentro do produto é definida de forma crescente segundo a identificação numérica dos itens. Vale ressaltar que a ordem dos itens dentro dos produtos finais é fixa.

Note que a quantidade de alelos utilizada da segunda parte é determinada pelo tamanho da lista de itens gerada. Considerando uma lista de tamanho k, são utilizados os k alelos da segunda parte do cromossomo. Desta forma, a lista de itens é pareada com os alelos para então ser ordenada de forma crescente de acordo com os valores dos alelos. Assim, é obtida a ordem de inserção da *bottom-left*. Como os produtos são selecionados de acordo com uma estimativa de empacotamento, é possível que a alocação falhe e, neste caso, o produto de menor prioridade é removido antes de se repetir o processo. Caso a alocação seja bem sucedida, os alelos utilizados são descartados. Na Figura 31, é ilustrada como é a relação entre as duas partes do cromossomo. Supondo que, após a ordenação da primeira parte (Figura 31b) e a estimativa de comprimento gasto, foi determinado que os produtos vermelho e verde serão empacotados no primeiro *bin* (Figura 31c). Como são cinco itens no total para confeccionar estes dois produtos (três itens utilizados no produto vermelho e dois utilizados no produto verde), serão utilizados cinco alelos da segunda parte do cromossomo para decidir a ordem de alocação destes itens (Figura 31c). Vale ressaltar que, se o empacotamento for bem sucedido, os alelos utilizados serão descartados.

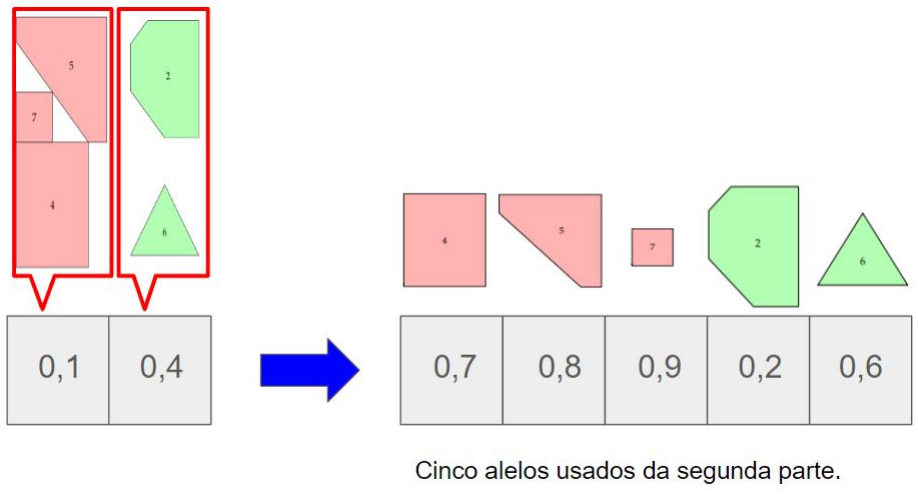
Figura 31 – Exemplo de cromossomo - Pareamento.



(a) Cromossomo completo.



(b) Ordenação da primeira parte.



(c) Pareamento dos itens.

Fonte: Elaborada pelo autor.

Como destacado anteriormente, Souza Queiroz e Andretta (2020) obtiveram melhores resultados ao permitir que algumas posições sejam ignoradas na heurística *bottom-left*, ou seja, nem sempre a posição mais abaixo e à esquerda é escolhida para alocar os itens. Para implementar tal técnica, as autoras adicionaram alelos que representavam a quantidade de posições que cada item ignora na regra *bottom-left*. Na Figura 32, estes alelos são representados pelo símbolo de proibido ao final do cromossomo. Desta forma, para cada vez que este símbolo aparece antes de um item, este item irá ignorar uma posição da *bottom-left*, ou seja, os itens 2 e 6 serão alocados na segunda posição disponível da *bottom-left*, pois o símbolo aparece uma vez antes de cada. Os demais itens serão alocados na primeira posição. Note que, se dois símbolos aparecessem antes do item 2, então ele seria alocado na terceira posição disponível, pois ele ignoraria duas posições disponíveis.

Figura 32 – Exemplo de cromossomo utilizado por Souza Queiroz e Andretta (2020).

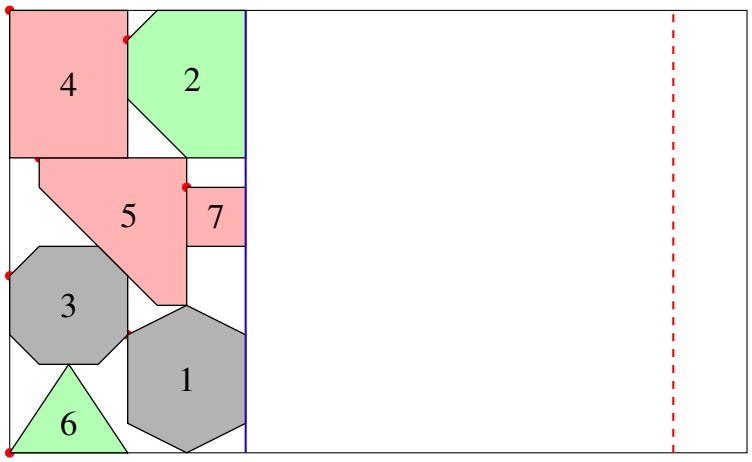
Fonte: Elaborada pelo autor.

Como Souza Queiroz e Andretta (2020) tratavam apenas um objeto de corte, é necessário adaptar tal técnica para a heurística deste trabalho. Desta forma, foram adicionados $s = \lfloor \beta * N_i \rfloor$ alelos no final do cromossomo, em que $\beta \in \mathbb{R}_+$ é um parâmetro a ser definido e N_i é a quantidade total de itens. Assim, na decodificação da segunda parte do cromossomo, para cada lista de alelos de tamanho k (lista de itens), também serão adicionados $k_{skip} = \lfloor \beta * k \rfloor$ alelos dos s, pois assim, em cada *bin*, haverá alelos responsáveis por ignorar posições. Juntando essas duas listas de alelos, é possível determinar a ordem de inserção utilizada pela *bottom-left* e também a quantidade de posições ignoradas pelos itens da mesma forma que foi exemplificado na Figura 32. Caso a alocação dos itens seja bem sucedida, os k_{skip} alelos utilizados são descartados, assim como

ocorre com os k alelos representando os itens.

O *fitness* de cada solução é dado pelo desperdício total de matéria-prima, ou seja, é calculado pela fórmula $\sum_{r\in\mathscr{O}}(L\, z^r_{max} - \sum_{w\in W} p_w\Delta^r_w)$, em que $\mathscr{O}$ é o conjunto de *bins* utilizados na solução, W é o conjunto de tipos de produtos finais, L é a altura do *bin*, z^r_{max} é o máximo entre o comprimento utilizado do *bin* r para empacotar todos os itens e o comprimento mínimo permitido para este *bin*, p_w é a área ocupada pelo produto w e Δ^r_w é a quantidade de produtos w que foram empacotados no *bin* r. Ou seja, a fórmula é a somatória dos desperdícios de matéria de cada *bin*. Note que não existem cromossomos infactíveis, pois todas as demandas são respeitadas e não são gerados planos de corte infactíveis. Na Figura 33, é ilustrado um exemplo de solução para a instância rco_2_90 considerando demandas mínimas e máximas de uma unidade. O produto 1 ocupa uma área de 30,0 unidades, enquanto os produtos 2 e 3 ocupam áreas de 41,0 e 24,3 unidades respectivamente. A linha azul indica o comprimento necessário para empacotar todos os produtos finais (8 unidades de comprimento), enquanto a linha vermelha indica o comprimento mínimo permitido para o *bin* (22,5 unidades). Como $z^r_{max} = \max\{8,\ 22,5\} = 22,5$, então segue que o *fitness* da solução é dado por $f = 15 * 22,5 - (30,0 + 41,0 + 24,3) = 242,2$.

Figura 33 – Exemplo de empacotamento feito para a instância rco_2_90.

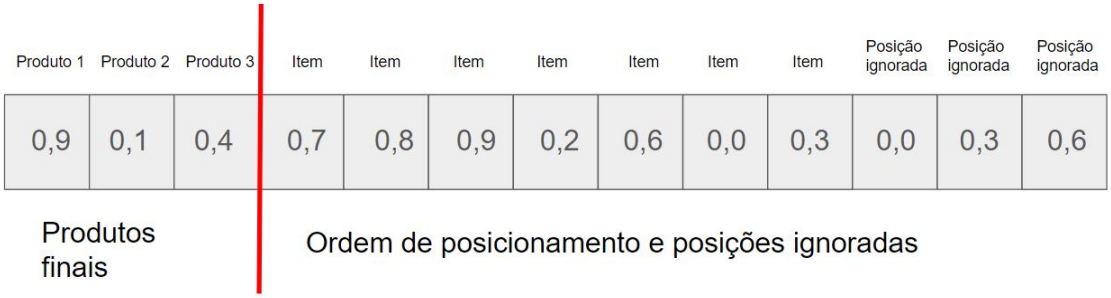


Fonte: Elaborada pelo autor.

Por exemplo, utilizando os produtos finais da instância rco_2 e considerando demandas mínimas e máximas de uma unidade para cada produto e $\beta = 0,5$, um possível cromossomo seria o ilustrado na Figura 34. Observe que existem três alelos para os produtos finais, sete alelos para os itens e $3 = \lfloor 0,5 * 7 \rfloor$ alelos para as posições ignoradas. Na Figura 35, é ilustrada a decodificação da primeira parte; note que a ordem de prioridade dos produtos é dada por 2, 3 e 1. Supondo que a estimativa de comprimento encontrou que é possível empacotar todos os produtos no *bin*, a lista de itens é composta pelos itens 4, 5, 7, 2, 6, 1 e 3. Logo, são utilizados sete alelos da segunda parte do cromossomo para o posicionamento. Além disso, são adicionadas $3 = \lfloor 0,5 * 7 \rfloor$ posições a serem ignoradas. Na Figura 36, é ilustrado como a segunda parte do cromossomo é decodificada; a ordem de inserção dos itens é 1, 2, 3, 6, 4, 5 e 7 e, além disso, os

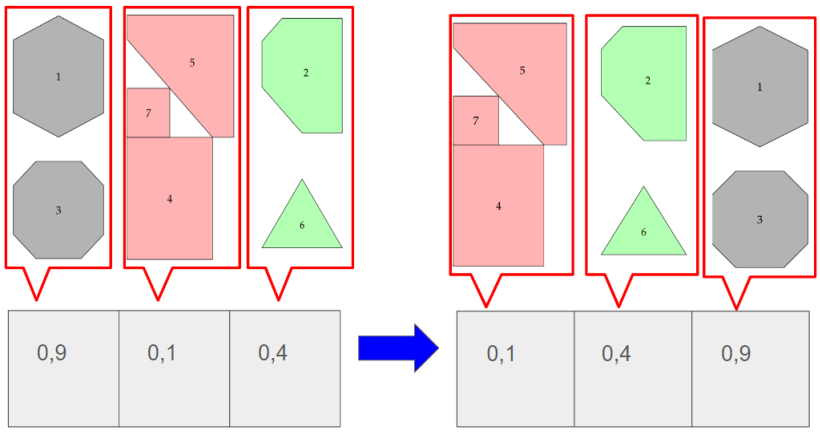
itens 2, 6 e 4 ignoram a primeira posição disponível. Caso seja possível empacotar todos os itens, as demandas mínimas serão satisfeitas e é calculado o *fitness* da solução. Caso contrário, será necessário empacotar menos produtos finais no *bin* e, consequentemente, outro plano de corte será gerado para atender as demandas mínimas. A Figura 37 ilustra o caso em que é possível empacotar todos os itens; assim, o *fitness* dessa solução é dado pelo desperdício de matéria-prima, sendo igual a 39,7 unidades de área.

Figura 34 – Exemplo de cromossomo completo.



Fonte: Elaborada pelo autor.

Figura 35 – Exemplo de decodificação da primeira parte.

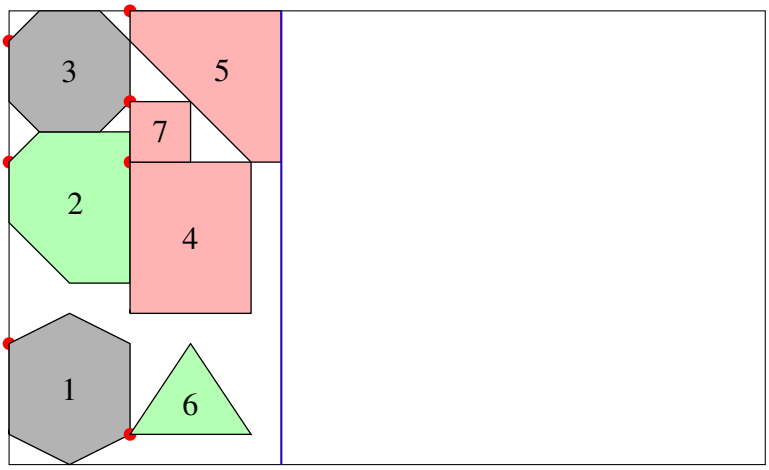


Fonte: Elaborada pelo autor.

Figura 36 – Exemplo de decodificação da segunda parte.

Fonte: Elaborada pelo autor.

Figura 37 – Empacotamento dos itens.



4.2 Experimentos computacionais

A heurística foi implementada em linguagem C++ e os testes foram realizados em um computador com processador Intel(R) Core(TM) i7-10700F CPU @ 2.90GHz, 16Gb de memória RAM e sistema operacional Ubuntu 22.04.2 LTS (64 bits), ou seja, o mesmo computador utilizado nos experimentos envolvendo o modelo matemático. Para avaliar o desempenho da heurística, foram utilizadas as mesmas instâncias da Subseção 3.2.1 considerando o comprimento mínimo igual a 90% *bin* (ou seja, $p = 90\%$). Ou seja, foram utilizadas as instâncias da subseção 3.2.3.

Os parâmetros da heurística foram calibrados utilizado a biblioteca *Iterated Race for Automatic Algorithm Configuration* (IRACE), desenvolvida por López-Ibáñez *et al.* (2016b). As instâncias utilizadas para a calibração foram rco_2_90, $blazewicz_2$_90 e $marques_2$_90, sendo imposto um tempo limite de 300 segundos para cada execução do BRKGA. Além disso, vale ressaltar que o número máximo de experimentos permitidos para a calibração foi de 1000 (ver mais detalhes em López-Ibáñez *et al.* (2016a)). Na Tabela 13, são resumidos os resultados obtidos na calibração. Um ponto importante a se destacar é que o IRACE não conseguiu provar a otimalidade dos parâmetros reportados por falta de iterações, entretanto, permitir mais iterações aumentaria significativamente o custo computacional do processo.

Tabela 13 – Parâmetros calibrados e resultados obtidos.

Parâmetro	Opções	Melhor opção
Tamanho da população (*Tam_pop*)	100, 200 ou 300	100
Probabilidade de herança do pai elite (*prob*)	0,6, 0,7 ou 0,8	0,6
Porcentagem de elites (*pe*)	0,1, 0,2 ou 0,3	0,3
Porcentagem de mutantes (*pm*)	0,05, 0,1 ou 0,15	0,15
Parâmetro de atualização da tabela *hash* (α)	0,1, 0,5 ou 0,9	0,9

Em relação ao parâmetro β (responsável por determinar a quantidade de posições ignoradas), foi utilizado o valor $\beta = 0,25$, pois este foi o mesmo valor que Souza Queiroz e Andretta (2020) adotaram. Dessa forma, é esperado reaproveitar a calibração feita pelas autoras. Vale ressaltar que as autoras também calibraram os parâmetros utilizando o IRACE.

Os experimentos foram divididos em quatro fases, sendo a primeira responsável por estudar o desempenho da heurística de forma geral. Na segunda fase, é proposto um método que busca melhorar o desempenho da heurística para as instâncias com mais de um *bin*. Na terceira fase, é estudado o uso de malhas mais refinadas (densas), sendo utilizada tanto a malha regular quanto a malha PBM. Por fim, na quarta, é estudada a capacidade da heurística de resolver instâncias-teste maiores. O critério de parada do BRKGA em todas as fases foi o tempo limite, sendo a execução finalizada quando o tempo de execução ultrapassar 3600 segundos.

4.2.1 Fase I

Na primeira fase, o objetivo é analisar a performance da heurística. Nesta subseção, é estudado o impacto que o uso da tabela *hash* apresenta na heurística e também se a técnica de ignorar posições proposta por Souza Queiroz e Andretta (2020) é adequada para o problema estudado. Para os testes desta fase, foram utilizadas malhas regulares idênticas às utilizadas nos experimentos da Subseção 3.2.3.

Para verificar se o uso da tabela *hash* é benéfico para a heurística, foram resolvidas as instâncias-testes sem utilizar a tabela *hash*. Na Tabela 14, são apresentados os resultados do

experimento. Na coluna "NG" é reportado o número de gerações até o término da execução, enquanto na coluna "NG_{evo}" é indicada a última geração em que ocorreu evolução (ou seja, houve melhora na solução). A coluna "T_{evo} (s)" relata o tempo decorrido até a última melhora na solução. As demais colunas são as mesmas que foram utilizadas nas tabelas da Seção 3.2.3, ou seja, a coluna "Instância" identifica a instância resolvida, "Nb" indica o número de *bins* considerados, "Min_c" é o comprimento mínimo que o *bin* pode assumir, "z^r" é o comprimento do *bin r* na solução obtida, "Desp. Total" é a somatória dos desperdícios de matéria-prima dos *bins*, "T (s)" é o tempo (em segundos) de resolução da instância, "Δ_{total}" é a quantidade total de itens empacotados, "Δ_w^r" indica a quantidade de cada produto final obtido em cada objeto de corte e "DpP" é o desperdício por produto final, ou seja, a razão dos desperdício total e a quantidade de produtos finais.

Tabela 14 – Resultados - BRKGA sem utilizar a tabela *hash*.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ_w^r	Δ_{total}	DpP	T_{evo} (s)	T (s)
rco_2_90	1	2483	0	22,5	[22,5]	130,2	[2,3,1]	6	21,7	0	3.601
rco_3_90		3678	0	22,5	[22,5]	119,8	[1,1,4]	6	20,0	0	3.601
rco_4_90		3662	0	22,5	[22,5]	129,7	[1,4,1]	6	21,6	0	3.600
rco_5_90		3360	0	22,5	[22,5]	129,7	[4,1,1]	6	21,6	0	3.600
rco_2_90	2	1845	6	22,5	[22,5; 22,5]	254,7	[3,3,0], [2,3,1]	12	21,2	11	3.602
rco_3_90		2166	19	22,5	[22,5; 22,5]	228,1	[1,0,5], [2,1,3]	12	19,0	30	3.601
rco_4_90		2004	1	22,5	[22,5; 22,5]	255,2	[1,4,1], [0,4,2]	12	21,3	3	3.600
rco_5_90		1816	6	22,5	[22,5; 22,5]	243,2	[4,0,2], [4,1,1]	12	20,3	10	3.601
rco_2_90	3	1301	0	22,5	[22,5; 22,5; 22,5]	395,9	[2,3,1], [3,2,1], [3,3,0]	18	22,0	2	3.602
rco_3_90		1574	20	22,5	[22,5; 22,5; 25,0]	373,6	[2,0,4], [3,1,2], [3,3,2]	20	18,7	46	3.601
rco_4_90		1684	19	22,5	[22,5; 22,5; 22,5]	410,9	[0,3,3], [1,3,2], [1,2,3]	18	22,8	42	3.601
rco_5_90		1346	18	22,5	[22,5; 22,5; 22,5]	374,9	[2,1,3], [3,0,3], [3,1,2]	18	20,8	41	3.601
$blazewicz_2$_90	1	2490	0	22,5	[22,5]	157,7	[2,3,1]	6	26,3	0	3.600
$blazewicz_3$_90		3477	0	22,5	[22,5]	148,3	[1,1,4]	6	24,7	0	3.600
$blazewicz_4$_90		2747	0	22,5	[22,5]	164,2	[1,4,1]	6	27,4	0	3.601
$blazewicz_5$_90		3560	0	22,5	[22,5]	146,2	[4,1,1]	6	24,4	0	3.601
$blazewicz_2$_90	2	1783	6	22,5	[22,5; 22,5]	307,7	[2,3,1], [3,3,0]	12	25,6	11	3.601
$blazewicz_3$_90		1819	10	22,5	[22,5; 22,5]	285,6	[2,0,4], [1,1,4]	12	23,8	16	3.602
$blazewicz_4$_90		1705	7	22,5	[22,5; 22,5]	326,7	[0,4,2], [1,4,1]	12	27,2	11	3.601
$blazewicz_5$_90		2007	3	22,5	[22,5; 22,5]	284,7	[5,0,1], [3,1,2]	12	23,7	6	3.602
$blazewicz_2$_90	3	1247	9	22,5	[22,5; 22,5; 22,5]	467,9	[2,3,1], [3,2,1], [3,3,0]	18	26,0	21	3.602
$blazewicz_3$_90		1470	12	22,5	[22,5; 22,5; 22,5]	451,1	[3,2,1], [3,0,3], [2,0,4]	18	25,1	28	3.601
$blazewicz_4$_90		1203	8	22,5	[22,5; 22,5; 23,0]	448,8	[2,2,2], [0,3,3], [2,3,3]	20	22,4	19	3.601
$blazewicz_5$_90		1635	15	22,5	[22,5; 22,5; 22,5]	458,9	[4,0,2], [3,0,3], [1,2,3]	18	25,5	34	3.600
$marques_2$_90	1	654	6	306,0	[306,0]	10897,8	[6]	6	1816,3	39	3.601

Os resultados mostram que, sem a tabela *hash*, o BRKGA não consegue evoluir a solução, pois a geração em que ocorre a última evolução é frequentemente a geração 0 (ou seja, não ocorre melhora na solução). Mesmo nos casos em que ocorre evolução, a geração em que isso ocorre ainda é bastante inicial, o que mostra uma convergência prematura da heurística. Outro fator que indica a convergência prematura é o tempo até a última melhora, que sempre se mostra baixo em relação ao tempo total de execução. A convergência prematura ocorre pois o comprimento necessário para empacotar os produtos finais é significativamente menor que a estimativa feita ao somar o comprimento necessário para empacotá-los individualmente. Desta forma, quando o BRKGA se baseia apenas nessa estimativa, a quantidade de produtos finais designada a cada *bin* é significativamente menor que a quantidade máxima suportada, o que leva à soluções de baixa

ocupação.

Na Tabela 15, são reportados os resultados do BRKGA utilizando a tabela *hash*. A coluna "$Diff$" mostra a diferença entre o desperdício total de matéria prima da solução obtida pela heurística e a pelo método exato; valores positivos indicam que o BRKGA apresentou aumento no desperdício. A coluna "MR (%)" indica a melhora percentual que a solução do BRKGA apresentou em relação à melhor solução do método exato (ou seja, a melhora relativa) considerando $p = 90\%$; valores negativos indicam que ocorreu uma piora na solução. Vale destacar que a melhor solução do método exato para as instâncias do tipo *rco* e *blazewicz* foram obtidas ao utilizar a malha regular 1x1, enquanto que a melhor solução para a instância *marques* foi obtida utilizando a malha *piece based mesh* (PBM).

Tabela 15 – Resultados - BRKGA - Malha regular - Com posições ignoradas.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
rco_2_90	1	978	222	22,5	[23,0]	44,7	[1,6,1]	8	5,6	13,3	-42,4	764	3.602
rco_3_90		1.048	966	22,5	[23,0]	49,1	[2,5,3]	10	4,9	16,5	-50,6	3.302	3.601
rco_4_90		1.340	37	22,5	[24,0]	39,7	[1,7,1]	9	4,4	5,4	-15,7	103	3.601
rco_5_90		1.272	517	22,5	[22,5]	55,2	[5,1,2]	8	6,9	19,1	-52,9	1.320	3.602
rco_2_90	2	536	334	22,5	[23,0; 22,5]	125,9	[3,4,1], [3,4,1]	16	7,9	41,1	-48,5	2.224	3.601
rco_3_90		573	484	22,5	[22,5; 24,0]	129,4	[2,4,3], [2,3,4]	18	7,2	43,5	-50,6	3.053	3.605
rco_4_90		786	105	22,5	[22,5; 23,0]	126,8	[1,4,3], [3,3,3]	17	7,5	48,4	-61,7	436	3.600
rco_5_90		727	691	22,5	[23,0; 23,0]	127,4	[3,1,4], [3,1,4]	16	8,0	53,0	-71,2	3.427	3.601
rco_2_90	3	97	19	22,5	[22,5, 22,5, 22,5]	250,1	[3,3,2], [2,2,4], [3,3,2]	24	10,4	0	0,0	697	3624
rco_3_90		144	142	22,5	[22,5; 22,5; 22,5]	250,1	[3,2,3], [3,1,3], [2,5,2]	24	10,4	0,0	0,0	3.586	3.611
rco_4_90		156	18	22,5	[22,5; 22,5; 22,5]	250,1	[2,3,3], [3,3,2], [3,2,3]	24	10,4	0,0	0,0	426	3.616
rco_5_90		148	16	22,5	[22,5; 22,5; 22,5]	250,1	[2,3,3], [4,2,2], [2,3,3]	24	10,4	0,0	0,0	392	3.618
$blazewicz_2$_90	1	679	169	22,5	[23,0]	75,2	[5,3,1]	9	8,4	19,4	-34,8	795	3.601
$blazewicz_3$_90		930	26	22,5	[23,0]	75,4	[4,4,2]	10	7,5	19,9	-35,9	86	3.600
$blazewicz_4$_90		1.325	887	22,5	[24,0]	78,1	[3,2,5]	10	7,8	24,4	-45,4	2.364	3.600
$blazewicz_5$_90		1.320	421	22,5	[24,0]	70,8	[5,4,1]	10	7,1	15,7	-28,5	1.130	3.601
$blazewicz_2$_90	2	484	180	22,5	[22,5; 24,0]	178,5	[2,4,2], [4,3,3]	18	9,9	45,0	-33,7	1.284	3.605
$blazewicz_3$_90		513	270	22,5	[24,0; 22,5]	184,5	[4,2,3], [2,6,2]	19	9,7	57,6	-45,4	1.863	3.602
$blazewicz_4$_90		694	239	22,5	[24,0; 23,0]	173,1	[6,0,4], [2,4,3]	19	9,1	48,9	-39,4	1.231	3.604
$blazewicz_5$_90		698	547	22,5	[22,5; 23,0]	167,4	[3,3,3], [5,4,0]	18	9,3	42,5	-34,0	2.797	3.601
$blazewicz_2$_90	3	131	0	22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [2,3,4], [4,2,1]	24	13,9	0,0	0,0	0	3.618
$blazewicz_3$_90		141	71	22,5	[22,5; 22,5; 22,5]	334,1	[1,5,3], [3,3,2], [4,0,3]	24	13,9	0,0	0,0	1.849	3.623
$blazewicz_4$_90		138	0	22,5	[22,5; 22,5; 22,5]	334,1	[1,4,3], [2,2,4], [5,2,1]	24	13,9	0,0	0,0	0	3.625
$blazewicz_5$_90		141	0	22,5	[22,5; 22,5; 22,5]	334,1	[3,2,3], [3,2,3], [2,4,2]	24	13,9	0,0	0,0	0	3.603
$marques_2$_90	1	470	6	306,0	[306,0]	10897,8	[6]	6	1816,3	2759,7	-33,9	34	3.605

Os resultados apresentados na Tabela 15 mostram que o uso da Tabela *hash* é relevante, pois nota-se que não há mais uma convergência prematura na maioria das instâncias. Além disso, é possível observar um aumento significativo na qualidade das soluções em relação às obtidas pela heurística sem a tabela *hash*. Entretanto, também é possível observar que a heurística perde em qualidade para o método exato em todas as instâncias, exceto nas que consideram três *bins*, pois estas são resolvidas na otimalidades tanto pelo BRKGA quanto pelo método exato por serem triviais. A perda em desempenho da heurística pode ser observada devido aos valores negativos na coluna "MR" (note que ocorreu uma piora percentual média de 28,98%) e também ao valores positivos na coluna "$Diff$".

Com o intuito de verificar a eficiência da técnica proposta por Souza Queiroz e Andretta (2020) para o nosso problema, também foram feitos testes que utilizaram $\beta = 0$, ou seja, o caso em que a heurística nunca ignora uma posição de alocação (*bottom-left* puro). A Tabela 16 mostra os resultados obtidos e a Tabela 17 mostra uma comparação entre as melhoras relativas

que ambas as versões da heurística obtiveram, sendo o melhor resultado destacado em negrito. A coluna "MP-CS" mostra melhora relativa da versão que ignora posições, enquanto a coluna "MP-SS" mostra a melhora relativa da versão que não ignora posições. Os resultados mostram que a versão que não ignora posições obtém resultados melhores ou iguais em todas as instâncias-teste, o que dá indícios de que ignorar posições não é vantajoso neste problema ou também que o valor β utilizado foi inadequado para o problema. Porém, mesmo obtendo melhores resultados, a heurística ainda não é capaz de atingir a qualidade das soluções obtidas pelo método exato, pois obtém uma piora percentual média de 21,27%.

Tabela 16 – Resultados - BRKGA - Malha regular - Sem posições ignoradas.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
rco_2_90	1	630	36	22,5	[23,0]	44,7	[1,6,1]	8	5,6	13,3	-42,4	152	3.603
rco_3_90		1.093	107	22,5	[23,0]	45,4	[4,4,2]	10	4,5	12,8	-39,3	365	3.602
rco_4_90		1.589	18	22,5	[24,0]	39,7	[1,7,1]	9	4,4	5,4	-15,7	37	3.600
rco_5_90		1.620	1466	22,5	[25,0]	52,1	[3,3,4]	10	5,2	16,0	-44,3	3.271	3.601
rco_2_90	2	532	239	22,5	[23,0; 23,0]	109,1	[4,3,2], [2,5,1]	17	6,4	24,3	-28,7	1.584	3.602
rco_3_90		774	634	22,5	[23,0; 23,0]	114,1	[1,4,4], [2,3,4]	18	6,3	28,2	-32,8	2.962	3.603
rco_4_90		845	193	22,5	[24,0; 24,0]	100,0	[1,8,0], [4,0,6]	19	5,3	21,6	-27,6	833	3.603
rco_5_90		914	392	22,5	[24,0; 25,0]	113,6	[5,0,3], [3,3,4]	18	6,3	39,2	-52,7	1.532	3.604
rco_2_90	3	447	2	22,5	[22,5; 22,5; 22,5]	250,1	[3,3,2], [4,2,2], [1,3,4]	24	10,4	0,0	0,0	20	3.600
rco_3_90		628	7	22,5	[22,5; 22,5; 22,5]	250,1	[3,3,2], [2,3,3], [3,2,3]	24	10,4	0,0	0,0	41	3.601
rco_4_90		748	26	22,5	[22,5; 22,5; 22,5]	250,1	[2,2,4], [4,3,1], [2,3,3]	24	10,4	0,0	0,0	115	3.603
rco_5_90		757	5	22,5	[22,5; 22,5; 22,5]	250,1	[3,1,3], [4,2,2], [1,5,3]	24	10,4	0,0	0,0	26	3.600
$blazewicz_2$_90	1	927	599	22,5	[23,0]	72,7	[4,4,1]	9	8,1	16,9	-30,3	2.216	3.603
$blazewicz_3$_90		1.038	100	22,5	[24,0]	70,9	[4,5,2]	11	6,4	15,4	-27,7	330	3.602
$blazewicz_4$_90		1.439	1349	22,5	[25,0]	69,0	[5,1,5]	11	6,3	15,3	-28,5	3.363	3.601
$blazewicz_5$_90		1.712	191	22,5	[23,0]	68,9	[5,2,2]	9	7,7	13,8	-25,0	411	3.601
$blazewicz_2$_90	2	504	397	22,5	[23,0; 24,0]	168,7	[4,3,2], [4,2,4]	19	8,9	35,2	-26,4	2.834	3.606
$blazewicz_3$_90		699	280	22,5	[22,5; 24,0]	158,3	[3,5,2], [5,3,2]	20	7,9	31,4	-24,7	1.410	3.602
$blazewicz_4$_90		708	515	22,5	[24,0; 24,0]	158,6	[4,3,3], [4,2,4]	20	7,9	34,4	-27,7	2.641	3.603
$blazewicz_5$_90		839	361	22,5	[22,5; 25,0]	155,1	[4,3,2], [3,5,3]	20	7,8	30,2	-24,2	1.556	3.601
$blazewicz_2$_90	3	434	2	22,5	[22,5; 22,5; 22,5]	334,1	[3,3,2], [1,4,4], [4,1,2]	24	13,9	0,0	0,0	21	3.602
$blazewicz_3$_90		602	30	22,5	[22,5; 22,5; 22,5]	334,1	[3,2,3], [2,3,3], [3,3,2]	24	13,9	0,0	0,0	171	3.602
$blazewicz_4$_90		717	7	22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [1,3,4], [5,2,1]	24	13,9	0,0	0,0	35	3.605
$blazewicz_5$_90		717	3	22,5	[22,5; 22,5; 22,5]	334,1	[4,0,3], [1,3,4], [3,5,1]	24	13,9	0,0	0,0	17	3.602
$marques_2$_90	1	457	8	306,0	[306,0]	10897,8	[6]	6	1816,3	2759,7	-33,9	57	3.607

Tabela 17 – Comparação entre as versões que ignoram ou não posições.

Instância	Nb	MP-CS MR (%)	MP-SS MR (%)
rco_2_90	1	-42,4	-42,4
rco_3_90		-50,6	**-39,3**
rco_4_90		-15,7	-15,7
rco_5_90		-52,9	**-44,3**
rco_2_90	2	-48,5	**-28,7**
rco_3_90		-50,6	**-32,8**
rco_4_90		-61,7	**-27,6**
rco_5_90		-71,2	**-52,7**
rco_2_90	3	0,0	0,0
rco_3_90		0,0	0,0
rco_4_90		0,0	0,0
rco_5_90		0,0	0,0
$blazewicz_2$_90	1	-34,8	**-30,3**
$blazewicz_3$_90		-35,9	**-27,7**
$blazewicz_4$_90		-45,4	**-28,5**
$blazewicz_5$_90		-28,5	**-25,0**
$blazewicz_2$_90	2	-33,7	**-26,4**
$blazewicz_3$_90		-45,4	**-24,7**
$blazewicz_4$_90		-39,4	**-27,7**
$blazewicz_5$_90		-34,0	**-24,2**
$blazewicz_2$_90	3	0,0	0,0
$blazewicz_3$_90		0,0	0,0
$blazewicz_4$_90		0,0	0,0
$blazewicz_5$_90		0,0	0,0
$marques_2$_90		-33,9	-33,9

4.2.2 Fase II

Como visto na fase anterior, o BRKGA não foi capaz de obter soluções tão boas quanto o método exato. Desta forma, nesta fase é proposta uma técnica para melhorar as soluções do BRKGA em instâncias com mais de um *bin*. A técnica se baseia no problema de cobertura, em que é necessário cobrir os pontos demandados ao mesmo tempo que se minimiza o gasto de recursos. Neste caso, o objetivo é cobrir todas as demandas mínimas de produtos finais utilizando planos de corte previamente obtidos ao longo da execução do BRKGA. Vale ressaltar que a somatória dos produtos finais devem respeitar suas respectivas demandas máximas e o recurso minimizado é a matéria-prima desperdiçada. Note que é possível obter planos de corte ao longo das iterações do BRKGA, pois cada vez que uma nova solução é decodificada, são gerados planos de corte (*bins*). Dessa forma, é possível utilizá-los como padrões para um modelo de cobertura.

Este problema de cobertura pode ser modelado da seguinte forma:

Conjuntos

- $\mathbb{P}$ é o conjunto de planos de corte previamente obtidos;
- W é o conjunto de tipos de produtos a serem confeccionadas (produtos finais).

Parâmetros

- $d_w^{\min}$ e $d_w^{\max}$ são os limitantes inferiores e superiores para a demanda de cada tipo de produto final $w \in W$;
- Nb é o número de planos de corte (número de *bins*);
- U_p^w é a quantidade de produtos finais do tipo $w \in W$ que o plano de corte $p \in \mathbb{P}$ gera;
- D_p é o desperdício de matéria-prima que o plano de corte $p \in \mathbb{P}$ gera.

Variáveis

- x_p é uma variável inteira que indica a quantidade de planos de corte do tipo $p \in \mathbb{P}$ utilizada.

$$\min \quad \sum_{p \in \mathbb{P}} D_p x_p \tag{4.1}$$

s.a

$$d_w^{\min} \leq \sum_{p \in \mathbb{P}} U_p^w x_p \leq d_w^{\max}, \quad \forall\, w \in W, \tag{4.2}$$

$$\sum_{p \in \mathbb{P}} x_p = Nb, \tag{4.3}$$

$$x_p \in \mathbb{N}, \qquad \forall p \in \mathbb{P}. \tag{4.4}$$

A função objetivo do modelo (4.1) minimiza o desperdício de matéria-prima. O conjunto de restrições (4.2) garante que a cobertura satisfaça todas as demandas mínimas e também que não ultrapasse as demandas máximas. A restrição (4.3) impõe que o número de planos de corte utilizados seja o mesmo que o exigido, ou seja, o número de *bins*. As restrições (4.4) representam o domínio das variáveis.

Note que é possível armazenar todos os planos de corte gerados ao longo da execução do BRKGA, porém isso aumentaria o custo computacional do modelo desnecessariamente, pois uma grande parte dos planos seriam de baixa qualidade e, portanto, não seriam úteis na solução. Logo, foram armazenados os planos de corte cujo desperdício é até 10% pior que o desperdício do melhor plano de corte conhecido pelo BRKGA até o momento (ou seja, conforme a solução melhora, o método armazena planos melhores).

O modelo desenvolvido foi codificado em linguagem Julia (versão 1.7.3) utilizando a biblioteca JuMP e resolvido pelo solver Gurobi (v9.5.1). Os testes com o método de cobertura foram feitos para as instâncias com dois *bins* considerando os planos de corte resultantes da Fase I. Note que no caso de um único *bin*, o resultado seria o mesmo do BRKGA. Já para o caso com três *bins*, o método não tem utilidade, já que são instâncias triviais em que o método exato e o BRKGA encontraram a solução ótima.

As Tabelas 18 e 19 mostram os resultados obtidos para o BRKGA que ignora posições e para a versão que não ignora posições respectivamente. A coluna "$Diff_{BRKGA}$" mostra a diferença de desperdício entre a solução obtida pelo método de cobertura e a solução do BRKGA (valores negativos indicam que a solução do método de cobertura é melhor). Já a coluna "MR_{BRKGA}" é a melhora relativa entre a solução do método da cobertura e a solução do BRKGA (valores positivos indicam que o método de cobertura obteve uma melhor solução).

Os resultados apontam que o método de cobertura consegue melhorar a solução na maioria das instâncias, porém as soluções ainda perdem em qualidade para as soluções do método exato. Assim como na primeira fase, os resultados do BRKGA que não ignora posições foram melhores, mostrando novamente que utilizar a regra *bottom-left* pura é mais vantajoso. Um ponto válido a se ressaltar é que, mesmo não sendo possível obter melhoria em todos os casos, utilizar o método de cobertura após o término do BRKGA é vantajoso, pois o tempo de execução do método é próximo de 0 segundo. Além disso, caso a solução do método de cobertura seja inferior à obtida pelo BRKGA, basta descartá-la e utilizar a de maior qualidade. Isso ocorre na instância rco_4_90, sendo causado possivelmente pela baixa tolerância na aceitação de planos de corte menos eficientes. Neste caso, os planos mais eficientes não podem ser utilizados em conjunto, pois as demandas máximas seriam violadas. Entretanto, vale ressaltar que aumentar a tolerância de aceitação também aumentaria o custo computacional do modelo, pois existiriam mais planos de corte.

Tabela 18 – Resultados - Cobertura por padrões - Malha regular - Com posições ignoradas.

Instância	Nb	Desp. Total	$Diff_{BRKGA}$	MR_{BRKGA} (%)	$Diff$	MR (%)	T (s)
rco_2_90	2	113,0	-12,9	10,2	28,2	-33,3	0
rco_3_90		115,4	-14,0	10,8	29,5	-34,3	0
rco_4_90		131,1	4,3	-3,4	52,7	-67,2	0
rco_5_90		111,4	-16,0	12,6	37,0	-49,7	0
$blazewicz_2$_90	2	150,8	-27,7	15,5	17,3	-13,0	0
$blazewicz_3$_90		150,2	-34,3	18,6	23,3	-18,4	0
$blazewicz_4$_90		154,2	-18,9	10,9	30,0	-24,2	0
$blazewicz_5$_90		151,8	-15,6	9,3	26,9	-21,5	0

Tabela 19 – Resultados - Cobertura por padrões - Malha regular - Sem posições ignoradas.

Instância	Nb	Desp. Total	$Diff_{BRKGA}$	MR_{BRKGA} (%)	$Diff$	MR (%)	T (s)
rco_2_90	2	101,9	-7,2	6,6	17,1	-20,2	0
rco_3_90		100,1	-14,0	12,3	14,2	-16,5	0
rco_4_90		135,3	35,3	-35,3	56,9	-72,6	0
rco_5_90		85,8	-27,8	24,5	11,4	-15,3	0
$blazewicz_2$_90	2	145,4	-23,3	13,8	11,9	-8,9	0
$blazewicz_3$_90		142,2	-16,1	10,2	15,3	-12,1	0
$blazewicz_4$_90		129,6	-29,0	18,3	5,4	-4,3	0
$blazewicz_5$_90		145,7	-9,4	6,1	20,8	-16,7	0

Tabela 20 – Resultados - BRKGA - Malha regular - Com posições ignoradas - Teste de ocupação.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
rco_2_90	1	931	232	22,5	[23,0]	44,7	[1,6,1]	8	5,6	13,3	-42,4	874	3.602
rco_3_90		1045	588	22,5	[22,5]	53,1	[1,6,3]	10	5,3	20,5	-62,9	1.915	3.604
rco_4_90		935	35	22,5	[24,0]	39,7	[1,7,1]	9	4,4	5,4	-15,7	124	3.602
rco_5_90		886	392	22,5	[22,5]	55,2	[5,1,2]	8	6,9	19,1	-52,9	1.531	3.601
$blazewicz_2$_90	1	961	94	22,5	[23,0]	75,2	[5,3,1]	9	8,4	19,4	-34,8	314	3.603
$blazewicz_3$_90		978	370	22,5	[23,0]	75,4	[4,4,2]	10	7,5	19,9	-35,9	1.305	3.600
$blazewicz_4$_90		971	13	22,5	[23,0]	65,8	[4,1,5]	10	6,6	12,1	-22,5	42	3.602
$blazewicz_5$_90		935	399	22,5	[23,0]	75,9	[4,2,3]	9	8,4	20,8	-37,7	1.515	3.602
$marques_2$_90	1	141	2	306,0	[306,0]	10897,8	[6]	6	1816,3	2759,7	-33,9	56	3.626

Tabela 21 – Resultados - BRKGA - Malha regular - Sem posições ignoradas - Teste de ocupação.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
rco_2_90	1	681	28	22,5	[23,0]	44,7	[1,6,1]	8	5,6	13,3	-42,4	130	3.602
rco_3_90		1058	384	22,5	[25,0]	45,4	[1,10,2]	13	3,5	12,8	-39,3	1.311	3.602
rco_4_90		999	13	22,5	[24,0]	39,7	[1,7,1]	9	4,4	5,4	-15,7	40	3.604
rco_5_90		989	570	22,5	[25,0]	51,6	[4,3,3]	10	5,2	15,5	-42,9	2.111	3.602
$blazewicz_2$_90	1	746	595	22,5	[23,0]	68,3	[3,3,4]	10	6,8	12,5	-22,4	2.905	3.600
$blazewicz_3$_90		933	227	22,5	[25,0]	70,6	[4,4,3]	11	6,4	15,1	-27,2	794	3.604
$blazewicz_4$_90		971	58	22,5	[23,0]	65,8	[4,1,5]	10	6,6	12,1	-22,5	201	3.600
$blazewicz_5$_90		902	481	22,5	[23,0]	68,9	[5,2,2]	9	7,7	13,8	-25,0	1.882	3.601
$marques_2$_90	1	668	8	306,0	[306,0]	10897,8	[6]	6	1816,3	2759,7	-33,9	38	3.604

4.2.3 Fase III

Uma alternativa para buscar a melhoria do desempenho da heurística é refinar a malha, pois a complexidade de memória exigida pelo BRKGA é significativamente menor que a exigida pelo método exato. Para refinar as malhas, foi duplicada sua resolução, ou seja, as malhas das instâncias do tipo *rco* e *blazewicz* passaram a ser 0,5x0,5, enquanto que a malha para a instância *marques* passou a ser 3,5x3,5. Além disso, para a instância *marques*, também foi utilizada a malha *piece based mesh* (PBM) proposta por Cherri *et al.* (2018), pois, como mostrado na Subseção 3.2.5, a malha PBM obtém melhores resultados para esse tipo de instância. No caso da malha PBM, foram utilizadas duas versões dela, sendo uma tendo os limites de refinamento igual a 7 (ou seja, $lim_x = lim_y = 7$) e a outra tendo os limites igual a 3,5 (ou seja, $lim_x = lim_y = 3,5$).

A Tabela 22 mostra os resultados do BRKGA (sem ignorar posições) para a malha regular, a Tabela 23 mostra os resultados para a malha PBM e a Tabela 24 mostra a comparação entre as melhoras relativas do BRKGA para cada malha, sendo o melhor resultado destacado em negrito. Espaços preenchidos com "X" indicam que não foram executados testes, ou seja, não foram feitos experimentos com a malha PBM para as instâncias do tipo *rco* e *blazewicz* pois resultados anteriores mostraram que esta malha não é adequada para estas instâncias (subseção 3.2.5).

Os resultados para a malha regular refinada (com mais pontos) foram piores ou iguais aos resultados para a malha padrão na maioria das instâncias do tipo *rco* e *blazewicz*, pois, como mostra a Tabela 24, a malha regular obtém resultados melhores ou iguais em todas as instâncias, com exceção da instância $blazewicz_2$_90 com dois *bins*. Porém foi melhor no caso da instância $marques_2$_90, em que a malha refinada obteve uma piora relativa de 24,9%, enquanto que a malha padrão obteve uma piora de 33,9%.

Em relação aos resultados da malha PBM, é possível observar que a versão com $lim_x = lim_y = 7$ obtém resultado melhor que a malha regular 7x7, porém pior que a malha regular refinada e a malha PBM utilizando $lim_x = lim_y = 3,5$, sendo esta a que apresenta o melhor resultado (apenas 2,5% pior que a solução do método exato e utilizando a metade do tempo). Vale ressaltar que em nenhum dos casos a heurística conseguiu obter resultados melhores que o método exato, o que é mostrado pelos valores negativos de MR.

A malha refinada ter um pior desempenho para as instâncias do tipo *rco* e *blazewicz* pode ser explicado pelo fato de que essas instâncias possuem itens cujos vértices são coordenadas inteiras em sua grande maioria. Dessa forma, a malha unitária facilita o encaixe dos itens nessas instâncias. Já para o caso da instância $marques_2$_90, os itens possuem encaixes mais difíceis, então a malha mais refinada e a PBM $lim_x = lim_y = 3,5$ acabam se mostrando superiores por permitir uma maior proximidade entre os itens.

Tabela 22 – Resultados - BRKGA - Malha regular refinada - Sem posições ignoradas.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
rco_2_90	1	202	76	22,5	[23,0]	44,7	[1,6,1]	8	5,6	13,3	-42,4	1.301	3.603
rco_3_90		217	161	22,5	[23,0]	51,3	[2,3,4]	9	5,7	18,7	-57,4	2.635	3.612
rco_4_90		219	16	22,5	[24,0]	39,7	[1,7,1]	9	4,4	5,4	-15,7	221	3.616
rco_5_90		215	51	22,5	[22,5]	55,2	[5,1,2]	8	6,9	19,1	-52,9	838	3.614
rco_2_90	2	119	38	22,5	[23,0; 22,5]	118,3	[4,3,2], [2,4,2]	17	7,0	33,5	-39,5	1.122	3.618
rco_3_90		122	105	22,5	[22,5; 24,5]	123,2	[1,4,4], [4,4,2]	19	6,5	37,3	-43,4	3.130	3.619
rco_4_90		135	92	22,5	[23,0; 24,0]	116,0	[3,2,4], [2,6,1]	18	6,4	37,6	-48,0	2.452	3.621
rco_5_90		121	57	22,5	[23,0; 25,0]	119,9	[3,1,4], [4,1,4]	17	7,1	45,5	-61,2	1.690	3.622
rco_2_90	3	100	3	22,5	[22,5; 22,5; 22,5]	250,1	[2,2,4], [1,4,3], [5,2,1]	24	10,4	0,0	0,0	127	3.606
rco_3_90		102	23	22,5	[22,5; 22,5; 22,5]	250,1	[3,2,3], [2,3,3], [3,3,2]	24	10,4	0,0	0,0	830	3.636
rco_4_90		104	3	22,5	[22,5; 22,5; 22,5]	250,1	[4,2,2], [2,3,3], [2,3,3]	24	10,4	0,0	0,0	127	3.612
rco_5_90		99	0	22,5	[22,5; 22,5; 22,5]	250,1	[4,2,2], [1,4,3], [3,2,3]	24	10,4	0,0	0,0	32	3.601
$blazewicz_2$_90	1	211	113	22,5	[23,5]	70,6	[5,2,3]	10	7,1	14,8	-26,5	1.936	3.612
$blazewicz_3$_90		212	60	22,5	[22,5]	74,6	[2,5,3]	10	7,5	19,1	-34,4	966	3.616
$blazewicz_4$_90		217	156	22,5	[23,5]	72,3	[4,2,4]	10	7,2	18,6	-34,6	2.575	3.607
$blazewicz_5$_90		213	190	22,5	[22,5]	69,1	[5,3,1]	9	7,7	14,0	-25,4	3.211	3.605
$blazewicz_2$_90	2	114	45	22,5	[23,0; 23,0]	173,5	[3,3,3], [4,3,2]	18	9,6	40,0	-30,0	1.419	3.619
$blazewicz_3$_90		113	45	22,5	[25,0; 23,0]	165,5	[4,2,4], [4,5,1]	20	8,3	38,6	-30,4	1.379	3.632
$blazewicz_4$_90		113	33	22,5	[24,5; 22,5]	169,4	[4,4,2], [3,2,4]	19	8,9	45,2	-36,4	1.014	3.614
$blazewicz_5$_90		111	36	22,5	[23,0; 23,0]	166,5	[4,3,2], [3,2,4]	18	9,2	41,6	-33,3	1.154	3.601
$blazewicz_2$_90	3	94	0	22,5	[22,5; 22,5; 22,5]	334,1	[3,3,2], [3,3,2], [2,2,4]	24	13,9	0,0	0,0	0	3.618
$blazewicz_3$_90		98	18	22,5	[22,5; 22,5; 22,5]	334,1	[2,3,3], [2,4,3], [4,1,2]	24	13,9	0,0	0,0	659	3.631
$blazewicz_4$_90		95	0	22,5	[22,5; 22,5; 22,5]	334,1	[2,2,4], [1,4,3], [5,2,1]	24	13,9	0,0	0,0	31	3.617
$blazewicz_5$_90		98	1	22,5	[22,5; 22,5; 22,5]	334,1	[4,2,2], [2,2,3], [2,4,3]	24	13,9	0,0	0,0	64	3.616
$marques_2$_90	1	40	13	306,0	[332,5]	10166,1	[7]	7	1452,3	2028,0	-24,9	1.306	3.655

Tabela 23 – Resultados - BRKGA - Malha PBM - Sem posições ignoradas.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	$Diff$	MR (%)	T_{evo} (s)	T (s)
$marques_2$_90 – $lim_x = lim_y = 7$	1	116	9	306,0	[332,5]	10634,1	[7]	7	1519,1	2496.0	-30,6	229	3.601
$marques_2$_90 – $lim_x = lim_y = 3,5$	1	40	11	306,0	[315,0]	8346,1	[7]	7	1192,3	208,0	-2,5	1.200	3.625

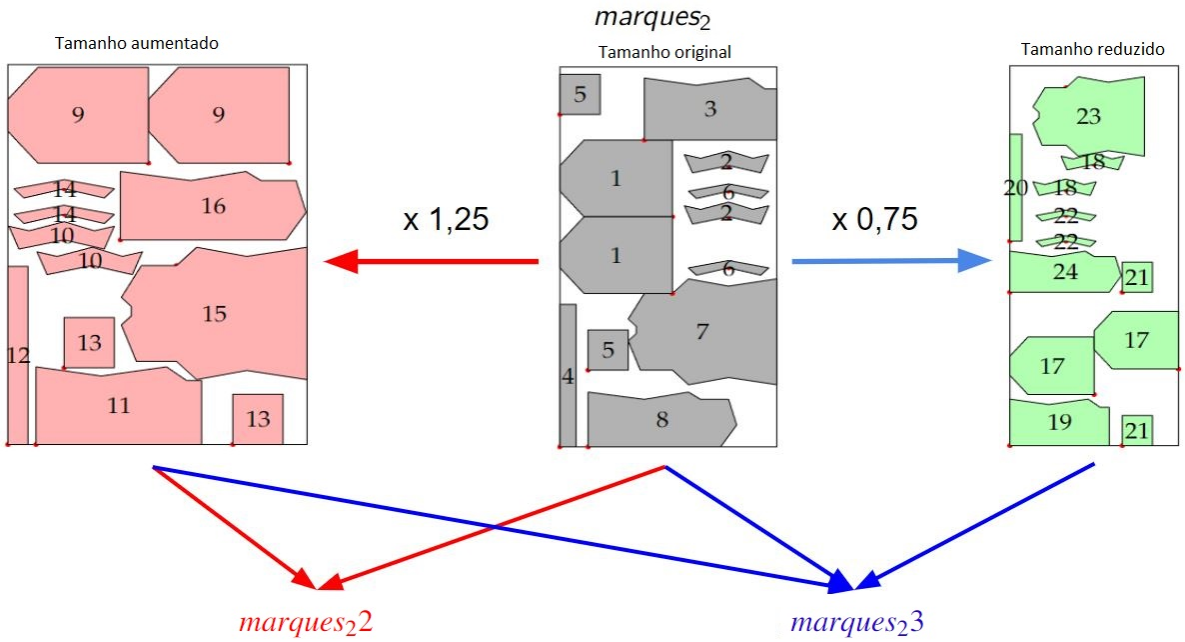
Tabela 24 – Comparação entre os resultados.

Instância	Nb	Malha regular padrão MR (%)	Malha regular refinada MR (%)	PBM ($lim_x = lim_y = 7$) MR (%)	PBM ($lim_x = lim_y = 3,5$) MR (%)
rco_2_90	1	-42,4	-42,4	X	X
rco_3_90		**-39,3**	-57,4		
rco_4_90		-15,7	-15,7		
rco_5_90		**-44,3**	-52,9		
rco_2_90	2	**-28,7**	-39,5	X	X
rco_3_90		**-32,8**	-43,4		
rco_4_90		**-27,6**	-48,0		
rco_5_90		**-52,7**	-61,2		
rco_2_90	3	0,0	0,0	X	X
rco_3_90		0,0	0,0		
rco_4_90		0,0	0,0		
rco_5_90		0,0	0,0		
$blazewicz_2$_90	1	-30,3	**-26,5**	X	X
$blazewicz_3$_90		**-27,7**	-34,4		
$blazewicz_4$_90		**-28,5**	-34,6		
$blazewicz_5$_90		**-25,0**	-25,4		
$blazewicz_2$_90	2	**-26,4**	-30,0	X	X
$blazewicz_3$_90		**-24,7**	-30,4		
$blazewicz_4$_90		**-27,7**	-36,4		
$blazewicz_5$_90		**-24,2**	-33,3		
$blazewicz_2$_90	3	0,0	0,0	X	X
$blazewicz_3$_90		0,0	0,0		
$blazewicz_4$_90		0,0	0,0		
$blazewicz_5$_90		0,0	0,0		
$marques_2$_90	1	-33,9	-24,9	-30,6	**-2,5**

4.2.4 Fase IV

A quarta fase tem como objetivo estudar o uso da heurística para resolver instâncias-teste maiores. Para isso, foram elaboradas duas instâncias a partir da instância $marques_2$. Como os itens de $marques_2$ formam uma camisa, é possível criar diferentes tamanhos deste produto ao escalar estes itens. Na Figura 38, é ilustrado o procedimento realizado para gerar as novas instâncias. Os itens da instância $marques_2$ foram escalados em 1,25 para criar uma camisa maior e em 0,75 para criar uma camisa menor. Dessa forma, a instância $marques_2 2$ contém as camisas de tamanho médio (original) e grande, enquanto a instância $marques_2 3$ as camisas de tamanho pequeno, médio e grande.

Figura 38 – Criação das novas instâncias.



Os resultados desta fase não puderam ser comparados com os resultados obtidos pelo modelo matemático, pois não foi possível resolver as novas instâncias utilizando o modelo. Note que a instância $marques_2 2$ possui dezesseis tipos de itens, enquanto a instância $marques_2 3$ possui vinte e quatro tipos de itens e, portanto, possuem um grande custo computacional tanto em relação ao tempo de execução quanto ao uso de memória.

As novas instâncias possuem demanda mínima de 1 e demanda máxima de 12 para cada produto final. A quantidade de *bins* variou de 1 até 3 e foi utilizado $p = 0,9$. A malha utilizada neste experimento foi a malha regular 7x7, uma vez que a malha PBM aumentaria o custo computacional.

São resumidos na Tabela 25 os resultados do BRKGA (sem ignorar posições) para as novas instâncias-teste. Dessa forma, o BRKGA é capaz de resolver instâncias maiores que o modelo matemático. Entretanto, também é possível observar que o número de gerações decorridas para as instâncias do tipo $marques_2 3$ é significativamente menor em comparação com o número

de gerações decorridas para as instâncias $marques_2 2$.

Tabela 25 – Resultados - BRKGA - Malha regular - Sem posições ignoradas.

Instância	Nb	NG	NG_{evo}	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	T_{evo} (s)	T (s)
$marques_2 2\_90$	1	191	182	306,0	[308,0]	10669,8	[3,2]	5	2134,0	3.477	3.615
$marques_2 2\_90$	2	103	47	306,0	[306,0; 306,0]	22013,6	[1,3], [0,4]	8	2751,7	1.727	3.625
$marques_2 2\_90$	3	68	44	306,0	[307,2; 307,2; 306,0]	34261,3	[0,4], [1,3], [1,3]	12	2855,1	2.324	3.632
$marques_2 3\_90$	1	80	22	306,0	[306,0]	10025,9	[1,3,1]	5	2005,2	1.031	3.636
$marques_2 3\_90$	2	56	36	306,0	[306,0; 307,0]	22117,6	[1,3,0], [1,3,1]	9	2457,5	2.388	3.642
$marques_2 3\_90$	3	37	31	306,0	[306,0; 307,0; 306,0]	36721,1	[0,3,2], [0,4,0], [1,1,4]	15	2448,1	3.160	3.633

Em uma tentativa de melhorar as soluções obtidas, foi aplicado o método de cobertura para as instâncias com 2 ou 3 *bins*. Os resultados obtidos são resumidos na Tabela 26. Note que a qualidade das novas soluções aumentou em todos os casos, indicando mais uma vez que o método de cobertura cumpre sua proposta.

Tabela 26 – Resultados - Cobertura por padrões - Malha regular - Sem posições ignoradas.

Instância	Nb	Min_c	z^r	Desp. Total	Δ^r_w	Δ_{total}	DpP	T (s)
$marques_2 2\_90$	2	306,0	[327,0; 306,0]	20709,9	[2,3], [0,4]	9	2301,1	0
$marques_2 2\_90$	3	306,0	[306,0; 307,2; 307,2]	32299,5	[1,3], [0,4], [0,4]	12	2691,6	0
$marques_2 3\_90$	2	306,0	[306,5; 300,0]	20103,8	[1,3,1], [0,4,0]	9	2233,8	0
$marques_2 3\_90$	3	306,0	[308,0; 315,0; 315,0]	28669,9	[3,2,0], [0,4,1], [0,4,1]	15	1911,3	0

4.3 Considerações

Neste capítulo, foi proposta uma heurística BRKGA para o problema estudado, pois resolvê-lo por meio da modelagem matemática se mostrou computacionalmente caro. Para auxiliar a heurística, foi implementada uma tabela *hash* que armazena uma estimativa de comprimento necessário para empacotar um conjunto de produtos finais. Dessa forma, a heurística utiliza a tabela para designar cada produto final a um *bin* de forma mais precisa. Experimentos computacionais mostraram que o uso da tabela impede que o BRKGA convirja de forma prematura e, portanto, seu uso se mostrou bastante importante na heurística. Além disso, os experimentos computacionais mostraram que ignorar posições neste problema não é efetivo, sendo mais vantajoso utilizar a heurística *bottom-left* pura para alocar os itens. Entretanto, vale ressaltar que o valor de β utilizado neste trabalho foi o mesmo que Souza Queiroz e Andretta (2020) e, portanto, pode não ser adequado para o problema estudado.

Em relação à qualidade das soluções obtidas, o BRKGA obteve soluções piores que o método exato em todas as instâncias, pois, ao longo de todos os experimentos, os valores de "MR" foram negativos (exceto no caso das instâncias com três *bins*). Para aumentar seu desempenho, foi proposta uma estratégia de cobertura das demandas por padrões de corte. Essa abordagem se mostrou eficaz, sendo capaz de melhorar a solução na maioria dos casos e tendo baixo custo computacional. Entretanto, este método é dependente da obtenção de bons planos de corte pelo BRKGA e, como foi visto, isso não ocorre.

Uma vantagem da heurística em relação ao método exato é seu baixo custo de memória. Dessa forma, é possível utilizar malhas mais refinadas para resolver as instâncias. Para as instâncias do tipo *rco* e *blazewicz*, o uso da malha refinada se mostrou pior, pois os encaixes dos itens são fortemente favorecidos pela malha unitária. Entretanto, para a instância *marques*, o uso da malha PBM mais refinada se mostrou mais eficiente, uma vez que os itens desta instância são mais difíceis de se encaixar e, portanto, a malha mais refinada possui vantagem ao permitir uma maior proximidade entre os itens. Entretanto, não foi possível obter resultados melhores que os obtidos pelo método exato.

Outra vantagem do BRKGA é sua capacidade de resolver instâncias que o método exato não consegue devido ao elevado custo computacional tanto em relação à memória quanto ao tempo de execução.

Um problema persistente na heurística desenvolvida é a sua dificuldade em obter soluções de alta qualidade considerando apenas um *bin*. Utilizando o método de cobertura, é possível obter soluções de qualidade maior para instâncias de mais de um *bin*, porém, para o bom funcionamento desta estratégia, é necessário que o BRKGA encontre planos de corte com baixo desperdício. Uma alternativa para aumentar o desempenho da heurística é utilizar outra regra de posicionamento. Por exemplo, Souza Queiroz e Andretta (2020) utilizaram as heurísticas *alternate bottom-left* e *alternate zigzag* além da *bottom-left*. Os experimentos feitos pelas autoras mostraram que utilizar a *alternate-horizontal-zig-zag* é mais vantajoso que utilizar a *bottom-left* para o problema. Outros exemplos de regra de posicionamento podem ser encontrados em Mundim *et al.* (2018).

CAPÍTULO

5

CONCLUSÕES

Este trabalho teve como objetivo estudar o problema de empacotamento de itens irregulares em *bins* considerando produtos finais. O problema foi inspirado nas indústrias têxteis e de confecção, nas quais os itens devem ser empacotados em conjunto em cada *bin* de modo que formem produtos finais. Este aspecto é importante por dois motivos: a) permitir que ao término do corte de um *bin*, o processo de costura dos produtos finais possa ser iniciado (assim evitando um gargalo na cadeia de produção); e b) assegurar a padronização dos produtos finais, evitando possíveis diferenças de tonalidade no tecido dos itens que compõem um mesmo produto.

As contribuições deste trabalho foram três: i) foi mostrado que o empacotamento dos itens considerando a formação de produtos finais é relevante, pois caso contrário não são obtidos produtos ao fim da etapa de corte do *bin*, ii) foi desenvolvido um modelo modelo matemático de programação inteira-mista para o problema e iii) foi proposta uma heurística para a resolução do problema.

O modelo matemático foi baseado no modelo de Toledo *et al.* (2013), pois este é o modelo mais eficiente quando se trata de empacotar itens de tipos repetidos em grandes quantidades. Nesse modelo, cada *bin* é representado por uma malha regular de pontos nos quais os itens podem ser posicionados. Quanto mais refinada (maior a quantidade de pontos) é a malha, maior é a chance de obter uma solução de boa qualidade, porém isso também aumenta o custo computacional do modelo, sendo assim necessário encontrar um equilíbrio entre o refinamento da malha e o tempo computacional necessário para resolver o problema.

Os experimentos computacionais mostraram que considerar o empacotamento em conjunto é bastante impactante, pois caso não seja considerado, raramente são obtidos produtos finais ao término do corte de um *bin*. Foi verificado também que o custo computacional de resolver o modelo é elevado, pois não foi encontrada uma solução ótima para a maioria das instâncias. Além disso, foi observado que permitir uma diminuição no comprimento do *bin* auxilia a criação de planos de corte mais eficientes, uma vez que, em determinadas situações,

não é possível alocar mais produtos finais no *bin*. Entretanto, o comprimento mínimo permitido deve ser escolhido com cuidado, pois utilizar planos de corte menores implica numa menor utilização da mesa de corte, o que acaba prejudicando a eficiência da linha de produção. Por fim, foi estudado o uso da malha *piece based mesh* (PBM) proposta por Cherri *et al.* (2018). Os resultados mostraram que esta malha é mais eficiente para a instância *marques*, pois para ela, não é possível utilizar uma malha regular refinada, uma vez que o custo computacional de memória para esta instância impede seu uso.

Devido à elevada complexidade computacional de resolução do modelo (tanto em relação à memória quanto ao tempo de execução para se obter uma solução ótima), não foi possível provar a otimalidade da maioria das soluções encontradas. Dessa forma, foi proposto um método heurístico para resolver o problema, visando encontrar boas soluções com um baixo tempo de execução. Foi desenvolvida uma a meta-heurística BRKGA (Gonçalves e Resende (2011)). Desta forma, o cromossomo é responsável por designar cada produto final a um *bin* e também por alocar cada item dentro de seu respectivo *bin*. A revisão bibliográfica mostrou que, segundo Souza Queiroz e Andretta (2020), ignorar algumas posições ao utilizar a regra de posicionamento *bottom-left* no BRKGA pode ser vantajoso e, portanto, esta estratégia foi inserida no BRKGA desenvolvido. Entretanto, os experimentos computacionais realizados mostraram que a heurística perde eficiência ao utilizar essa técnica, sendo melhor utilizar a regra *bottom-left* pura para posicionar os itens. Para melhorar o desempenho da heurística, foi desenvolvido um método de cobertura por padrões e também estudado o uso de malhas mais refinadas, porém nenhuma dessas alternativas foi capaz de fazer a heurística alcançar os resultados obtidos pelo método exato. Entretanto, vale destacar que o BRKGA foi capaz de resolver instâncias maiores que não podem ser resolvidas pelo modelo desenvolvido.

Como trabalhos futuros, são sugeridos estudos relacionados a *math*-heurísticas envolvendo o modelo desenvolvido. A desvantagem do método exato é seu elevado custo computacional, entretanto, resolver instâncias menores é relativamente rápido. Portanto, como *math*-heurísticas conseguem decompor o modelo matemático em partes para resolvê-lo, é interessante analisar o uso de tais heurísticas considerando o modelo, pois os resultados do método exato (mesmo não sendo provada a otimalidade em várias instâncias) se mostraram superiores quando comparados aos resultados obtidos pela heurística.

Com relação ao BRKGA, uma direção para trabalhos futuros é estudar outras regras de posicionamento assim como fizeram Pinheiro, Amaro-Junior e Saraiva (2016) e Souza Queiroz e Andretta (2020), pois como os experimentos mostraram, o BRKGA tem dificuldade em obter planos de corte eficientes. Outra opção para melhorar o desempenho do BRKGA é propor uma nova forma de realizar a decodificação dos cromossomos.

REFERÊNCIAS

ABIT. **Associação Brasileira da Indústria Têxtil e de Confecção - Perfil do Setor**. 2023. Acessado: 21/08/2023. Disponível em: <https://www.abit.org.br/cont/perfil-do-setor>. Citado na página 19.

ALSAMARAH, W.; YOUNES, B.; YOUSEF, M. Reducing waste in garment factories by intelligent planning of optimal cutting orders. **The Journal of The Textile Institute**, v. 113, p. 1917–1925, 2022. Citado nas páginas 19, 20, 24 e 52.

ALVAREZ-VALDÉS, R.; MARTINEZ, A.; TAMARIT, J. A branch & bound algorithm for cutting and packing irregularly shaped pieces. **International Journal of Production Economics**, v. 145, p. 463–477, 2013. Citado nas páginas 29 e 30.

ALVES, A. S. **Algoritmos para o encaixe de moldes com formato irregular em tecidos listrados**. Tese (Doutorado), Porto Alegre, RS, 2016. Citado nas páginas 30 e 53.

AMARO-JUNIOR, B.; PINHEIRO, P. R.; SARAIVA, R. D. A hybrid methodology for nesting irregular shapes: Case study on a textile industry. **IFAC Proceedings Volumes**, v. 46, p. 15–20, 2013. Citado na página 24.

BALDACCI, R.; BOSCHETTI, M. A.; GANOVELLI, M.; MANIEZZO, V. Algorithms for nesting with defects. **Discrete Applied Mathematics**, v. 163, p. 17–33, 2014. Citado nas páginas 29 e 30.

BENNELL, J. A.; OLIVEIRA, J. F. The geometry of nesting problems: A tutorial. **European Journal of Operational Research**, v. 184, p. 397–415, 2008. Citado nas páginas 24 e 30.

____. A tutorial in irregular shape packing problems. **Journal of the Operational Research Society**, 60, p. S93–S105, 2009. Citado nas páginas 19 e 39.

BISCHOFF, E. E.; RATCLIFF, M. Issues in the development of approaches to container loading. **Omega**, v. 23, p. 377–390, 1995. Citado na página 52.

BORTFELDT, A.; WÄSCHER, G. Constraints in container loading–a state-of-the-art review. **European Journal of Operational Research**, v. 229, p. 1–20, 2013. Citado na página 52.

CHEHRAZAD, S.; ROOSE, D.; WAUTERS, T. A fast and scalable bottom-left-fill algorithm to solve nesting problems using a semi-discrete representation. **European Journal of Operational Research**, v. 300, p. 809–826, 2022. Citado nas páginas 43 e 49.

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, p. 7085–7102, 2018. Citado nas páginas 48, 57, 59, 66, 67, 72, 93 e 100.

CHERRI, L. H.; MUNDIM, L. R.; ANDRETTA, M.; TOLEDO, F. M. B.; 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, p. 570–583, 2016. Citado nas páginas 25, 26, 29, 30, 31, 33, 35, 37 e 48.

DEAN, H. T. **Minimizing waste in the 2-dimensional cutting stock problem**. Tese (Doutorado), Christchurch, Nova Zelândia, 1996. Citado nas páginas 29 e 30.

DOWSLAND, K. A.; DOWSLAND, W. B. Solution approaches to irregular nesting problems. **European Journal of Operational Research**, v. 84, p. 506–521, 1995. Citado na página 19.

DYCKHOFF, H.; FINKE, U. **Cutting and packing in production and distribution: Typology and bibliography**. [S.l.]: Springer Science & Business Media, 1992. Citado nas páginas 20 e 23.

ELKERAN, A. A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. **European Journal of Operational Research**, v. 231, p. 757 – 769, 2013. Citado nas páginas 39, 40, 41, 43, 44, 45, 46, 47, 48 e 49.

ESICUP. **Working Group on Cutting and Packing within EURO**. 2021. Disponível em: <www.euro-online.org/websites/esicup/data-sets/#1535972088237-bbcb74e3-b507.Acesso: 18/11/2021.> Citado na página 57.

FISCHETTI, M.; LUZZI, I. Mixed-integer programming models for nesting problems. **Journal of Heuristics**, v. 15, p. 201–226, 2009. Citado nas páginas 29 e 30.

FOWLER, R. J.; PATERSON, M. S.; TANIMOTO, S. L. Optimal packing and covering in the plane are np-complete. **Information processing letters**, v. 12, p. 133–137, 1981. Citado nas páginas 21, 23 e 53.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. **Operations Research**, v. 9, p. 849–859, 1961. Citado na página 19.

____. A linear programming approach to the cutting stock problem — Part II. **Operations Research**, v. 11, p. 863–888, 1963. Citado na página 19.

GONÇALVES, J. F.; RESENDE, M. G. Biased random-key genetic algorithms for combinatorial optimization. **Journal of Heuristics**, v. 17, p. 487–525, 2011. Citado nas páginas 73 e 100.

HECKMANN, R.; LENGAUER, T. A simulated annealing approach to the nesting problem in the textile manufacturing industry. **Annals of Operations Research**, v. 57, p. 103–133, 1995. Citado nas páginas 23 e 24.

HEISTERMANN, J.; LENGAUER, T. The nesting problem in the leather manufacturing industry. **Annals of Operations Research**, v. 57, p. 147–173, 1995. Citado na página 23.

HONO, D. Y.; TOLEDO, F. M. O problema corte de produtos irregulares em bins da indústria de confecção. **Anais do Simpósio Brasileiro de Pesquisa Operacional**, v. 55, p. 160563, 2023. Citado nas páginas 51 e 58.

KIM, J. H.; LEE, J. Draft layout generation of building drawings on real urban scenes with boundary particle method and priority solver. **Multimedia Tools and Applications**, v. 80, p. 29539–29560, 2021. Citado na página 23.

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, p. 712–721, 2016. Citado nas páginas 26, 30, 33, 37 e 48.

LEÃO, A. A. S.; TOLEDO, F. M. B.; OLIVEIRA, J. F.; CARRAVILLA, M. A.; ALVAREZ-VALDÉS, R. Irregular packing problems: A review of mathematical models. **European Journal of Operational Research**, v. 282, p. 803–822, 2020. Citado nas páginas 19, 29 e 30.

LI, X.; WANG, Z.; CHAN, F. T.; CHUNG, S. H. A genetic algorithm for optimizing space utilization in aircraft hangar shop. **International Transactions in Operational Research**, v. 26, p. 1655–1675, 2019. Citado na página 23.

LÓPEZ-IBÁÑEZ, M.; CÁCERES, L. P.; DUBOIS-LACOSTE, J.; STÜTZLE, T. G.; BIRATTARI, M. **The irace package: User guide**. [S.l.]: IRIDIA, Institut de Recherches Interdisciplinaires et de Développements en . . . , 2016. Citado na página 86.

LÓPEZ-IBÁÑEZ, 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. ISSN 2214-7160. Citado na página 86.

LU, Z.; HU, K.; NG, T. S. Improving additive manufacturing production planning: A sub-second pixel-based packing algorithm. **Computers Industrial Engineering**, v. 181, p. 109318, 2023. Citado nas páginas 21, 39, 40, 41, 42, 49, 73 e 80.

LUO, Q.; RAO, Y. Improved sliding algorithm for generating no-fit polygon in the 2d irregular packing problem. **Mathematics**, v. 10, p. 2941, 2022. Citado na página 26.

____. Heuristic algorithms for the special knapsack packing problem with defects arising in aircraft arrangement. **Expert Systems with Applications**, v. 215, p. 119392, 2023. Citado na página 23.

MARTINEZ-SYKORA, A.; ALVAREZ-VALDÉS, R.; BENNELL, J.; RUIZ, R.; TAMARIT, J. Matheuristics for the irregular bin packing problem with free rotations. **European Journal of Operational Research**, v. 258, p. 440–455, 2017. Citado nas páginas 47, 48 e 49.

M'HALLAH, R.; BOUZIRI, A. Heuristics for the combined cut order planning two-dimensional layout problem in the apparel industry. **International Transactions in Operational Research**, v. 23, p. 321–353, 2016. Citado nas páginas 20, 23, 24 e 53.

MIRHASSANI, S.; BASHIRZADEH, A. J. A grasp meta-heuristic for two-dimensional irregular cutting stock problem. **The International Journal of Advanced Manufacturing Technology**, v. 81, p. 455–464, 2015. Citado nas páginas 42 e 49.

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, p. 709–732, 2018. Citado nas páginas 11, 40, 43, 44, 47, 49 e 98.

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 21, 29, 39, 42, 45, 49, 73 e 79.

National Geographic. **Fast fashion goes to die in the world's largest fog desert. The scale is breathtaking.** 2024. Acessado: 19/05/2024. Disponível em: <https://www.nationalgeographic.com/environment/article/chile-fashion-pollution>. Citado na página 19.

OLIVEIRA, A. H. R. D.; TODARO, M. E. C. Arranjo físico do sistema produtivo de uma fábrica de uniformes. **XXXIV ENEGEP. Curitiba**, 2014. Citado na página 51.

PINHEIRO, P. R.; AMARO-JUNIOR, B.; SARAIVA, R. D. A random-key genetic algorithm for solving the nesting problem. **International Journal of Computer Integrated Manufacturing**, v. 29, p. 1159–1165, 2016. Citado nas páginas 26, 39, 40, 42, 49 e 100.

PLANKOVSKYY, S.; TSEGELNYK, Y.; SHYPUL, O.; PANKRATOV, A.; ROMANOVA, T. Cutting irregular objects from the rectangular metal sheet. In: **Integrated Computer Technologies in Mechanical Engineering**. [S.l.: s.n.], 2020. p. 150–157. Citado na página 23.

QIN, Y.; CHAN, F. T.; CHUNG, S. H.; QU, T.; NIU, B. Aircraft parking stand allocation problem with safety consideration for independent hangar maintenance service providers. **Computers & Operations Research**, v. 91, p. 225–236, 2018. Citado na página 23.

RODRIGUES, M. O.; CHERRI, L. H.; MUNDIM, L. R. Mip models for the irregular strip packing problem: new symmetry breaking constraints. **ITM Web Conf.**, v. 14, p. 00005, 2017. Citado nas páginas 25, 33 e 34.

RODRIGUES, M. O.; TOLEDO, F. M. B. A clique covering mip model for the irregular strip packing problem. **Computers & Operations Research**, v. 87, p. 221–234, 2017. Citado na página 49.

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, p. 657–671, 2019. Citado nas páginas 26, 29, 41, 43, 44, 45, 46, 47, 48 e 49.

SATO, A. K.; MARTINS, T. de C.; TSUZUKI, M. S. G. A pairwise exact placement algorithm for the irregular nesting problem. **International Journal of Computer Integrated Manufacturing**, p. 1177–1189, 2016. Citado nas páginas 42 e 49.

SATO, A. K.; MUNDIM, L. R.; MARTINS, T. C.; TSUZUKI, M. S. G. A separation and compaction algorithm for the two-open dimension nesting problem using penetration-fit raster and obstruction map. **Expert Systems with Applications**, p. 119716, 2023. Citado nas páginas 26, 29, 43, 45, 46 e 49.

SCHEITHAUER, G.; TERNO, J. Modeling of packing problems. **Optimization**, v. 28, p. 63–84, 1993. Citado nas páginas 25, 29 e 30.

Souza Queiroz, L. R. d.; ANDRETTA, M. Two effective methods for the irregular knapsack problem. **Applied Soft Computing**, v. 95, p. 106485, 2020. Citado nas páginas 11, 21, 39, 40, 41, 42, 49, 73, 75, 80, 82, 86, 88, 97, 98 e 100.

TAKAHARA, S.; KUSUMOTO, Y.; MIYAMOTO, S. Solution for textile nesting problems using adaptive meta-heuristics and grouping. **Soft Computing**, v. 7, p. 154–159, 2003. Citado na página 24.

TECNOLOGIA-TREINAMENTO. **Aprenda a confeccionar uma camisa masculina com listras verticais**. 2024. Acessado: 16/02/2024. Disponível em: <https://encurtador.com.br/cehqG>. Citado na página 52.

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, p. 478–487, 2013. Citado nas páginas 26, 30, 33, 35, 37, 38, 42, 45, 48, 49, 53, 54, 55, 72 e 99.

TSAO, Y.-C.; DELICIA, M.; VU, T.-L. Marker planning problem in the apparel industry: Hybrid pso-based heuristics. **Applied Soft Computing**, v. 123, p. 108928, 2022. Citado na página 24.

TSAO, Y.-C.; VU, T.-L.; LIAO, L.-W. Hybrid heuristics for the cut ordering planning problem in apparel industry. **Computers & Industrial Engineering**, v. 144, p. 106478, 2020. Citado na página 24.

UMETANI, S.; MURAKAMI, S. Coordinate descent heuristics for the irregular strip packing problem of rasterized shapes. **European Journal of Operational Research**, 2022. Citado nas páginas 46 e 49.

WASCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European Journal of Operational Research**, v. 183, p. 1109–1130, 2007. Citado nas páginas 20 e 23.

ZHANG, H.; LIU, Q.; WEI, L.; ZENG, J.; LENG, J.; YAN, D. An iteratively doubling local search for the two-dimensional irregular bin packing problem with limited rotations. **Computers Operations Research**, v. 137, p. 105550, 2022. Citado nas páginas 48 e 49.


ICMC
USP
SÃO CARLOS