UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Ciências Aplicadas

CARLOS JULIAN MUÑOZ QUIROGA

UM ESTUDO SOBRE FERRAMENTAS COMPUTACIONAIS NA RESOLUÇÃO DE PROBLEMAS DE CORTE E EMPACOTAMENTO IRREGULAR

LIMEIRA
2024

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Ciências Aplicadas

CARLOS JULIAN MUÑOZ QUIROGA

UM ESTUDO SOBRE FERRAMENTAS COMPUTACIONAIS NA RESOLUÇÃO DE PROBLEMAS DE CORTE E EMPACOTAMENTO IRREGULAR

Dissertação apresentada à Faculdade de Ciências Aplicadas da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia de Produção e de Manufatura nas áreas de Pesquisa Operacional e Gestão de Processos.

Orientador: Prof. Dr. Washington Alves de Oliveira
Coorientadora: Profa. Dra. Petra Maria Bartmeyer

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO CARLOS JULIAN MUÑOZ QUIROGA, E ORIENTADA PELO PROF. DR. WASHINGTON ALVES DE OLIVEIRA.

LIMEIRA
2024

Ficha catalográfica
Universidade Estadual de Campinas (UNICAMP)
Biblioteca da Faculdade de Ciências Aplicadas
Ana Luiza Clemente de Abreu Valério - CRB 8/10669

Quiroga, Carlos Julian Muñoz, 1986-
Q484e Um estudo sobre ferramentas computacionais na resolução de problemas de corte e empacotamento irregular / Carlos Julian Muñoz Quiroga. – Limeira, SP : [s.n.], 2024.

Orientador(es): Washington Alves de Oliveira.
Coorientador(es): Petra Maria Bartmeyer.
Dissertação (mestrado) – Universidade Estadual de Campinas (UNICAMP), Faculdade de Ciências Aplicadas.

1. Modelos matemáticos. 2. Geometria combinatória. I. Oliveira, Washington Alves, 1977-. II. Bartmeyer, Petra Maria, 1990-. III. Universidade Estadual de Campinas (UNICAMP). Faculdade de Ciências Aplicadas. IV. Título.

Informações complementares

Título em outro idioma: A study on computational tools for solving irregular cutting and packing problems
Palavras-chave em inglês:
Mathematical models
Combinatorial geometry
Área de concentração: Pesquisa Operacional e Gestão de Processos
Titulação: Mestre em Engenharia de Produção e de Manufatura
Banca examinadora:
Washington Alves de Oliveira [Orientador]
Larissa Tebaldi de Oliveira
Aline Aparecida de Souza Leão
Data de defesa: 26-08-2024
Programa de Pós-Graduação: Engenharia de Produção e de Manufatura

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0009-0002-0519-5030
- Currículo Lattes do autor: http://lattes.cnpq.br/6003620260161717

Folha de Aprovação

Autor: CARLOS JULIAN MUÑOZ QUIROGA
Título: Um estudo sobre ferramentas computacionais na resolução de problemas de corte e empacotamento irregular
Natureza: Dissertação
Área de Concentração: Pesquisa Operacional e Gestão de Processos
Instituição: Faculdade de Ciências Aplicadas – FCA/Unicamp
Data da Defesa: Limeira-SP, 26 de agosto de 2024.

BANCA EXAMINADORA:

Prof. Dr. Washington Alves de Oliveira (Orientador)
Faculdade de Ciências Aplicadas - FCA/Unicamp

Profa. Dra. Aline Aparecida de Souza Leão (membro externo)
Departamento de Matemática - Universidade Estadual de Londrina

Profa. Dra. Larissa Tebaldi de Oliveira (membro mxterno)
Faculdade de Ciências - UNESP - Campus Bauru

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria do Programa da Unidade.

AGRADECIMENTOS

Primeiramente, agradeço a Deus pelas inúmeras bênçãos e por me conceder a oportunidade e a força para empreender este processo de realizar minha maestria. Sua orientação e presença constante foram fundamentais ao longo desta jornada. Gostaria também de expressar minha profunda gratidão a todos que contribuíram para a realização deste trabalho:

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001

A meu orientador, Prof. Dr. Washington Alves de Oliveira, por sua orientação, paciência e apoio durante todo o processo.

À minha coorientadora, Profa. Dra. Petra Maria Bartmeyer, por suas valiosas contribuições e suporte científico e técnico ao longo da pesquisa.

À minha família, pelo amor incondicional e incentivo constante.

Aos meus colegas e amigos, pela colaboração e momentos de descontração que tornaram esta jornada mais leve e agradável.

A todos que, de alguma forma, contribuíram direta ou indiretamente para a conclusão deste trabalho, expresso meu sincero agradecimento.

EPÍGRAFE

"A verdadeira viagem de descoberta não consiste em procurar novas paisagens, mas em ter novos olhos".

Marcel Proust

Resumo

Os problemas de empacotamento irregular são uma classe ampla de problemas de otimização que envolve a determinação do layout de um conjunto de peças menores irregulares sobre um contêiner, chamado de objeto, de tal forma que um layout final pode ser alcançado, por exemplo, ao minimizar o espaço ocioso no objeto. A complexidade combinatória envolvida na determinação do melhor layout é um desafio a ser superado nas metodologias de resolução para esses problemas. Adicionalmente, é preciso lidar com a dificuldade computacional de implementar cada detalhe geométrico para eliminar qualquer sobreposição entre peças e manter completamente cada peça no interior do objeto. Uma grande parte das metodologias de resolução combina sofisticados modelos matemáticos, incluindo a representação precisa da geometria, com técnicas de aprendizado de máquina, heurísticas e meta-heurísticas, além do uso de softwares avançados de resolução. Esta dissertação apresenta um estudo sobre ferramentas computacionais na resolução dos problemas de corte e empacotamento irregular en faixas e apresenta uma nova metodologia de representação do problema baseada na sua geometria, que forneceu suporte para elaborar um novo método de resolução e também pode ser usada na proposição de modelos matemáticos, bem como no desenvolvimento de novas técnicas de solução, para as aplicações práticas da indústria. Mais especificamente, o método de resolução proposto neste trabalho consiste de uma heurística bioinspirada na interação gravitacional de corpos e baseada em formulações de hipóteses de melhoria tal que a sua rotina evolui por meio da validação dessas hipóteses e da incorporação de algumas técnicas da literatura. Resumidamente, um pré-processamento ordena a sequência de posicionamento das peças de acordo com o valor decrescente das áreas, então uma solução inicial é construída posicionando as peças em grupos ou por camadas e, finalmente, esta solução é otimizada com o uso de uma metaheurística de recozimento simulado. O desempenho computacional também foi melhorado ao incluir uma estratégia de escolha dinâmica de orientações permitidas. A heurística de interação gravitacional teve bom desempenho na resolução de instâncias da literatura e a metodologia proposta de representação tem potencial para avanços.

Palavras-chaves: Empacotamento irregular, métodos bioinspirados

Abstract

Nesting problems are a broad class of optimization problems that involve determining the layout of a set of smaller irregular pieces within a container, referred to as the object, in such a way that a final layout can be obtained, for example, by minimizing idle space within the object. The combinatorial complexity involved in determining the best layout is a challenge to be overcome in the solution methodologies for these problems. Additionally, it is necessary to deal with the computational difficulty of implementing each geometric detail to eliminate any overlap between pieces and to completely keep each piece within the object. A large part of the solution methodologies combines sophisticated mathematical models, including precise geometry representation, with machine learning techniques, heuristics, and metaheuristics, as well as the use of advanced solution software. This dissertation presents a study on computational tools for solving irregular cutting and packing problems and introduces a new methodology for representing the problem based on its geometry. This methodology provided support for developing a new resolution method and can also be used in the formulation of mathematical models as well as in the development of new solution techniques for practical industrial applications. More specifically, the resolution method proposed in this work consists of a bio-inspired heuristic based on the gravitational interaction of bodies and on the formulation of improvement hypotheses such that its routine evolves through the validation of these hypotheses and the incorporation of some techniques from the literature. In summary, a pre-processing step orders the sequence of piece placements according to decreasing area values, then an initial solution is constructed by placing the pieces in groups or layers, and finally, this solution is optimized using a simulated annealing metaheuristic. Computational performance was also enhanced by including a strategy for the dynamic selection of permitted orientations. The gravitational interaction heuristic performed well in solving instances from the literature, and the proposed representation methodology has potential for further advancements.

Keywords: Nesting problem, bio-inspired methods

Lista de Figuras

1 Exemplo de layout para manufatura de peças de vestuário. Fonte: Bennell & Oliveira (2008) . . . 15

2 Representação raster 0-1 para peças irregulares. Fonte: Oliveira & Ferreira (1993) . . . 18

3 Representação discreta de partes e objeto contidos em um retângulo imaginário. Fonte: Babu & Babu (2001) . . . 19

4 Interpretação da D-função. Fonte: Bennell & Oliveira (2008) . . . 20

5 Traçando o movimento do polígono B em torno de A para formar o NFP. Fonte: Bennell & Oliveira (2008) . . . 20

6 Regra de posicionamento *bottom left*. Fonte: Bennell & Oliveira (2009) . . . 22

7 Solução viável e seu gráfico de conectividade. Fonte: Martins & Tsuzuki (2008) . . . 24

8 Exemplo de posicionamento de uma peça durante o processo de otimização. Fonte: Martins & Tsuzuki (2008) . . . 25

9 BRKGA para o problema nesting. Fonte: Mundim et al. (2017) . . . 26

10 Fluxograma geral do método de resolução proposto. Fonte: Própria . . . 34

11 Representação de peças por arranjo de coordenadas. Fonte: Própria . . . 35

12 Representação de peças por arranjo de parâmetros de linha reta das arestas e coordenadas dos vértices. Fonte: Própria . . . 36

13 Representação de peças por arranjo de parâmetros de reta e coordenadas. Fonte: Própria 36

14 Exemplo da regra de posicionamento IG. Fonte: Própria . . . 38

15 Fluxograma do método de resolução proposto incluindo as estratégias de otimização. Fonte: Própria . . . 41

16 Classificação das peças em relação aos limites do comprimento. Fonte: Própria . . . 42

17 Exemplo de compactação por camadas. Fonte: Própria . . . 45

18 Exemplo de posicionamento de peça em etapa de otimização com recozimento simulado. (a) Solução atual. (b) Retrocesso aleatório na sequência de posicionamento e início das fases do recozimento. (c) Mudança de peça de referência. (d) Fases do recozimento e restrições. Fonte: Própria . . . 47

19 Estrutura dos arranjos para estratégia de escolha dinâmica de orientações permitidas. (a) configuração dos arranjos, (b) exemplo de parametrização inicial. Fonte: Própria . 49

20 Eficiência e tempo computacional das melhores soluções obtidas pelos algoritmos IG e IGO para as instâncias originais. Fonte: Própria 54
21 Eficiência e tempo computacional das melhores soluções obtidas pelos algoritmos IG e IGO para as instâncias em RCO_n e $BLAZEWICZ_n$. Fonte: Própria 55
22 Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Ordem_Camadas. Fonte: Própria 56
23 Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Ordem_Camadas nos primeiros 60 segundos. Fonte: Própria 57
24 Eficiência no tempo dos algoritmos IGO_RS e IGO_Ordem_Camadas. Fonte: Própria 58
25 Eficiência no tempo dos algoritmos IGO_RS_OP e IGO_RS. Fonte: Própria 59

26 Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Camadas: a) no tempo limite de 1800 segundos; e b) nos primeiros 60 segundos. Fonte: Própria 71
27 Soluções da instância *Albano* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 72
28 Soluções da instância *Blaz* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 73
29 Soluções da instância *Jakobs* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 74
30 Soluções da instância *Jakobs2* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 75
31 Soluções da instância *Marques* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 76
32 Soluções da instância *Poly1a* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 77
33 Soluções da instância *Shirts* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria 78

Lista de Tabelas

4.1 Característica geral de cada instância 51
4.2 Resultados dos algoritmos IG e IGO para as instâncias originais 53
4.3 Resultados dos algoritmos IG e IGO para as instâncias RCO_n e $BLAZEWICZ_n$ 54
4.4 Comparação dos melhores resultados entre IGO_RS_OP e aqueles da literatura (retirados de Burke et al. (2010), Elkeran (2013), Peralta et al. (2018), Sato et al. (2019), Sato et al. (2023)) 60

Conteúdo

Introdução **14**

1 Fundamentação teórica **16**

1.1 Problemas de corte e empacotamento . . . 16

1.2 Ferramentas geométricas . . . 17

1.2.1 Raster . . . 17

1.2.2 Trigonometria direita . . . 18

1.2.3 Polígono *no-fit* (NFP) . . . 19

1.2.4 Função Phi . . . 21

1.3 Técnicas de resolução e otimização . . . 22

1.3.1 Metaheurísticas . . . 23

1.3.2 Aprendizado de máquina . . . 26

2 Revisão da literatura **29**

3 Descrição do problema e método de resolução proposto **33**

3.1 Descrição do problema . . . 33

3.2 Método de resolução . . . 34

3.2.1 Representação das peças . . . 35

3.2.2 Construção da solução inicial e otimização da solução . . . 35

3.2.3 Otimização da regra por hipóteses . . . 42

3.2.4 Regra de ordenação e estratégia de compactação . . . 43

3.2.5 Otimização com recozimento simulado . . . 46

3.2.6 Otimização com escolha dinâmica de orientações permitidas . . . 48

4 Testes computacionais **51**

4.1 Descrição das instâncias e desenho dos testes realizados . . . 51

4.2 Resultados para os algoritmos IG e IGO . . . 53

4.2.1 Desempenho em relação as instâncias originais . . . 53

4.2.2 Desempenho em relação as instâncias modificadas 54
4.3 Resultados para as modificações do algoritmo IGO nas instâncias originais 55
4.4 Síntese dos resultados computacionais 60

5 Conclusões **61**

Bibliografia **63**

Anexo 1: Modelos matemáticos **67**

Anexo 2: Teste adicional para a instância Blaz **71**

Anexo 3: Ilustração gráfica das soluções obtidas com o método proposto **72**

Introdução

Os problemas de corte e empacotamento são parte natural de muitas aplicações industriais e de logística que aparecem em formato próprio ou incorporados em outros problemas de otimização mais abrangentes. Eles compartilham uma estrutura geométrica singular cuja as muitas configurações diferentes e características específicas definem uma ampla variedade de problemas.

Uma classe abrangente de problemas de corte e empacotamento inclui aos problemas de empacotamento irregular, conhecidos em inglês por *nesting problems*. Cada um desses problemas envolve o posicionamento geométrico (layout) de um conjunto de peças menores (irregulares e regulares) sobre um contêiner (objeto grande) retangular ou irregular, de tal forma que um layout final é alcançado, por exemplo, ao minimizar o espaço ocioso no objeto (sobra de material). Para garantir a factibilidade do problema, é necessário que não haja sobreposição entre as peças e que todas as peças estejam completamente no interior do objeto (Leão et al. 2020). Estas duas classes de restrições formam o maior desafio na resolução do problema, pois a complexidade do conjunto de restrições vai depender dos tipos de peças envolvidas. O aspecto combinatório inerente a todos os problemas de corte e empacotamento, combinado às restrições de factibilidade para peças irregulares, faz deste um problema de difícil resolução. De fato, o esforço computacional envolvido cresce exponencialmente com o número de peças e depende do tipo de representação geométrica escolhida para as peças e objeto. Uma grande parte das metodologias de resolução combina sofisticados modelos matemáticos com técnicas heurísticas e metaheurísticas, além do uso de softwares avançados de resolução.

Exemplos de aplicação para o problema de empacotamento irregular aparecem na indústria de vestuário, móveis, couro, vidro e corte de chapas metálicas. A Figura 1 ilustra um exemplo típico de solução para um problema de empacotamento irregular encontrado em indústrias de vestuário, no qual é considerado um conjunto de peças com formas irregulares que são empacotadas por meio de uma configuração factível (todas as peças no interior do objeto e sem sobreposição entre elas) sobre uma placa retangular. O exemplo da figura refere-se a um problema de empacotamento em faixas, no qual apenas duas dimensões da matéria prima principal são relevantes no processo de corte. Enquanto a largura do objeto é fixa, o comprimento do objeto (faixa) pode ser considerado como uma variável do problema, ou seja, o objetivo é cortar todas as peças minimizando o comprimento usado do objeto. Note ainda que boas soluções encontradas carregam naturalmente economia de matéria prima e baixo

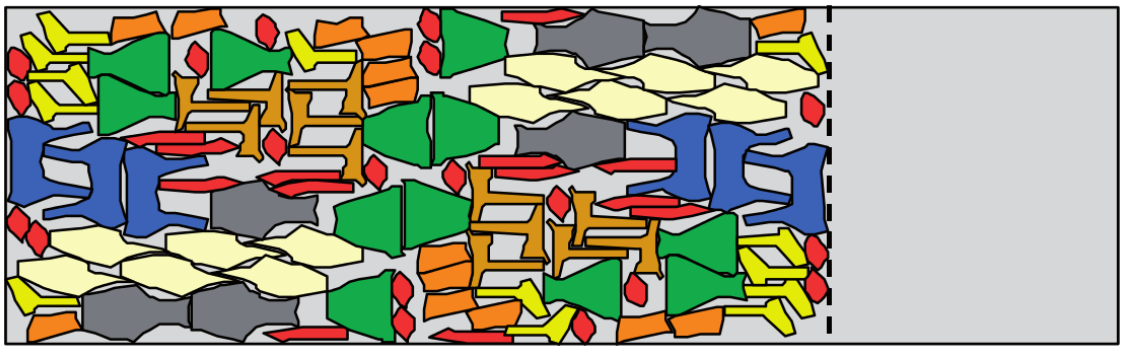


Figura 1: Exemplo de layout para manufatura de peças de vestuário. Fonte: Bennell & Oliveira (2008)

impacto ambiental.

Adicionalmente, dependendo da prática industrial no processo de corte, restrições tecnológicas ou características específicas adicionais devem ser usadas para empacotar cada peça, incluindo restrições sobre a orientação das peças (Rocha et al. 2014). Por exemplo, na indústria moveleira e têxtil, as orientações permitidas estão geralmente restritas a alguns ângulos de orientação (orientações discretas), enquanto em outras indústrias que lidam com o corte de chapas de couro ou folhas de metal, quaisquer orientações são aceitas (orientações livres). Desta forma, a seleção de uma representação geométrica adequada pode reduzir ou aumentar a dificuldade para a metodologia de resolução escolhida, uma vez que cada representação escolhida carrega limitações próprias, como, por exemplo, a possibilidade de rotação angular parcial das peças.

A contribuição central deste trabalho é propor uma técnica de resolução para o problema de corte e empacotamento de peças com geometria irregular, em faixas de material de largura fixa e comprimento variável, a partir do estudo de ferramentas geométricas e computacionais. A técnica proposta visa combinar métodos computacionais de otimização com regras que permitam delimitar o conjunto de soluções do problema. Ela consiste numa regra de acomodação (posicionamento) consecutiva de peças isentas de sobreposição entre elas e limitadas ao interior de um contêiner retangular ou faixa, considerando um cálculo de distância mínima de colisão como estratégia de verificação de sobreposição. A formulação de hipóteses e a incorporação adequada de regras e outras técnicas encontradas na literatura, converte a técnica proposta numa heurística de resolução robusta. Para validar a heurística de resolução proposta foram realizados testes computacionais sobre instâncias encontrados na literatura e feitas comparações com resultados de outros estudos.

O restante do texto é dividido da seguinte maneira. O Capítulo 1 descreve os principais conceitos teóricos que ajudam a compreender o problema e suportam as técnicas de resolução mais utilizadas. O Capítulo 2 apresenta uma breve revisão da literatura relacionada às principais formas de representação e resolução do problema de corte e empacotamento irregular. O Capítulo 3 descreve o problema estudado e apresenta o método de resolução proposto neste trabalho, explicando detalhadamente as rotinas e técnicas utilizadas. O Capítulo 4 apresenta os resultados obtidos com o método de resolução proposto e as principais características da sua implementação. Finalmente, o Capítulo 5 resume os principais resultados encontrados e apresenta propostas para trabalhos futuros.

CAPÍTULO 1

Fundamentação teórica

Este capítulo apresenta a fundamentação teórica que define os conceitos e dá suporte para a compreensão dos métodos de resolução utilizados no problema de corte e empacotamento, detalhando um conjunto de definições do problema que inclui tipologia, ferramentas geométricas e técnicas de resolução. Também foram revisadas técnicas que utilizam modelagem matemática para resolver problemas de empacotamento, algumas das quais são apresentadas no Anexo 5. Apesar de este trabalho não implementar diretamente uma técnica de resolução baseada em um modelo matemático, é crucial conhecê-las, considerando que estudos futuros poderiam explorar formulações baseadas na modelagem matemática do problema a partir da abordagem proposta.

1.1 Problemas de corte e empacotamento

O problema de corte é uma das aplicações clássicas da programação inteira, cuja a ideia geral é determinar como um objeto maior pode ser cortado em peças menores de diferentes e variados tamanhos para satisfazer demandas ou ordens de clientes, ao mesmo tempo que é minimizada a perda do material. Similarmente, o problema de empacotamento aloca peças menores em um objeto de maior tamanho, buscando maximizar o espaço utilizado. Os problemas de corte e empacotamento aparecem naturalmente incorporados no processo de produção da indústria têxtil, de móveis, de aço, etc. Além disso, em termos da geometria das aplicações, o problema pode ser caracterizado como unidimensional, quando uma dimensão é relevante no processo de corte, como exemplos são os cortes de barras de aço, bobinas ou rolos de papel, bidimensional, quando duas dimensões são relevantes no processo de corte, como nos cortes de placas de madeira, tecidos ou chapas de aço (Arenales et al. 2007), ou tridimensional, se como no caso da fabricação de peças em impressão 3D, as três dimensões são relevantes (Yau & Hsu 2022).

Para classificar os problemas de corte e empacotamento, diferentes tipologias têm sido propostas na literatura. Dyckhoff & Finke (1992) apresentaram a primeira tipologia conhecida, na qual foram utilizadas quatro características para classificar os problemas: a dimensionalidade do objeto e das peças; o tipo de atribuição das peças e dos objetos, que se refere a se o problema consiste em alterar as

dimensões do objeto para que todas as peças possam caber, ou em alterar a seleção das peças para que caibam dentro do objeto; a quantidade e variedade de objetos, que podem ser um único objeto ou vários objetos idênticos ou diferentes; e a variedade de peças, que podem ser idênticas (sem variedade), peças semelhantes (de mesmo formato e dimensões diferentes) e peças de diferentes formatos. A classificação aprimorada proposta por Wäscher et al. (2007) é uma extensão da primeira tipologia e é descrita na sequência.

1. **Dimensionalidade:** número de dimensões do problema estudado. O problema pode ser unidimensional, bidimensional ou tridimensional. Problemas com mais de três dimensões são classificados como extensões do problema de corte e empacotamento.

2. **Tipo de atribuição:** o objetivo pode ser classificado como maximização da saída ou minimização da entrada. Para problemas de maximização da produção, os objetos não são suficientes para acomodar todas as peças. Para problemas de minimização da entrada, todas as peças devem ser alocadas minimizando o custo.

3. **Variedade de peças:** as peças podem ser idênticas, com um grau de similaridade ou diferentes.

4. **Variedade de objetos:** permite um objeto com todas as dimensões fixas, um objeto com uma ou mais dimensões abertas e vários objetos com dimensões fixas, que podem ser idênticos, similares ou diferentes.

5. **Formato de peças:** classifica as peças bidimensionais e tridimensionais em regulares e irregulares. Exemplos de peças regulares são retângulos, círculos, caixas, cilindros e esferas. As peças irregulares ou não regulares são polígonos convexos e não convexos que podem conter buracos.

1.2 Ferramentas geométricas

Uma questão importante nos problemas de corte e empacotamento é a capacidade de lidar com a estrutura geométrica derivada da não regularidade e não convexidade das peças irregulares. Este problema consiste da necessidade de obter posicionamentos viáveis para todas as peças, ou seja, as peças precisam ser colocadas dentro de uma faixa ou objeto que também pode ser irregular. Por esse motivo, surgiu a necessidade de desenvolver ferramentas e representações geométricas eficientes e robustas (Gomes 2013). Diferentes ferramentas geométricas foram propostas na literatura para abordar o problema de empacotamento de peças irregulares, sendo que as mais conhecidas são exemplificadas abaixo.

1.2.1 Raster

Os métodos raster ou pixel têm como principal característica dividir a área contínua de um objeto em áreas discretas, permitindo reduzir a informação geométrica para codificar os dados em uma grade representada por uma matriz. Um esquema de codificação simples utiliza o valor 1 para codificar a existência da peça e o valor 0 para identificar o espaço vazio, como ilustrado na Figura 2.

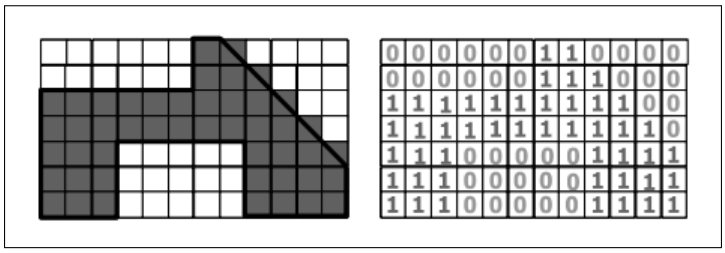


Figura 2: Representação raster 0-1 para peças irregulares. Fonte: Oliveira & Ferreira (1993)

A forma de alocar uma peça no objeto consiste em adicionar zeros e uns que representam cada peça, na matriz que representa o layout. Para uma determinada posição do layout, o valor do raster correspondente a entrada da matriz é a quantidade de peças que ocupam a posição, então se este valor for maior que 1 significa que existe sobreposição (Oliveira & Ferreira 1993). A Figura 3 ilustra outro esquema de codificação que foi proposto por Babu & Babu (2001), considerando uma peça retangular, uma peça com arestas ortogonais e uma peça irregular com uma orientação particular. Devido à dificuldade para considerar qualquer referência para a disposição das peças na faixa, é feita a determinação do envoltório retangular das peças. O envoltório retangular de uma peça é o menor retângulo que envolve todos os vértices da peça no plano, de forma que cada lado do retângulo esteja alinhado paralelamente com um dos eixos coordenados. Em seguida, o envoltório retangular é dividido numa grade uniforme de 1mm $\times$ 1mm de tamanho. Cada elemento na grade refere-se a um pixel. Um valor específico é atribuído a cada pixel com base na sua posição em relação à geometria da peça. Este valor é 0, se o pixel estiver parcial ou totalmente na peça, caso contrário, este valor é considerado como um valor inteiro positivo. Este último começando em 1, para o pixel mais à direita, e incrementando uma unidade à medida que se move para o pixel mais à esquerda do envoltório retangular. Para peças com opção de rotação, o procedimento de representação descrito acima é seguido para cada ângulo de rotação.

Para a codificação do objeto os valores da matriz são zero se o pixel está inteiramente dentro do objeto. Caso contrário, considera-se um valor inteiro positivo crescente começando da direita para a esquerda, como se observa na Figura 3. A simplicidade de codificação é uma vantagem dos métodos pixel. Além disso, eles permitem calcular a distância que uma peça deve ser deslocada para eliminar uma infactibilidade, ou ainda, conhecer onde alocar duas peças adjacentes em contato uma com a outra, bastando contar a quantidade de pixels na direção desejada. No entanto, os métodos pixel não podem representar exatamente peças com arestas não ortogonais e, muitas vezes, necessitam de memória computacional relativamente alta para funcionar corretamente.

1.2.2 Trigonometria direita

Os métodos de trigonometria direita são utilizados para lidar eficientemente com a natureza irregular da geometria dos problemas de corte e empacotamento, utilizando diretamente uma representação vetorial para cada parte. Neste método, a factibilidade de um posicionamento fica implícita e é capaz de identificar colisões. A trigonometria direita envolve teste de interseção de segmentos de reta e inclusão de pontos. Embora os testes sejam mais complexos em termos computacionais em comparação

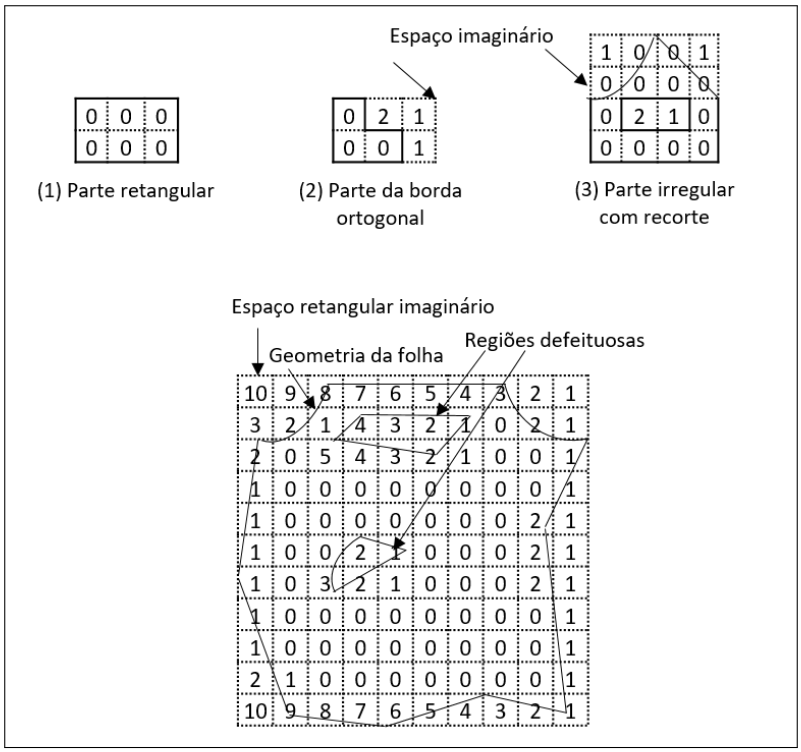


Figura 3: Representação discreta de partes e objeto contidos em um retângulo imaginário. Fonte: Babu & Babu (2001)

aos métodos raster, o tempo de execução não pode ser comparado diretamente, porque para o raster o tempo para verificar a factibilidade é quadrático no tamanho da grade e, para a trigonometria direita, é exponencial no número de arestas dos polígonos (Elkeran 2013).

Uma das estratégias mais conhecidas da trigonometria direita envolve o uso das D-funções. A D-função, proposta inicialmente por Konopasek (1981), é uma estratégia que fornece a posição relativa de um ponto P em relação a uma aresta orientada AB baseando-se na equação de distância entre um ponto a um segmento de reta, tal que, esse segmento de reta é limitado e pode ser chamado como linha de suporte para a aresta. Uma D-função pode ser definida no plano Euclidiano com origem no ponto $(0, 0)$ da seguinte forma. Seja $D_{ABP} = ((X_A - X_B)(Y_A - Y_P) - (Y_A - Y_B)(X_A - X_P))$ a relação numérica entre o ponto P e a aresta suporte AB, onde X e Y são coordenadas dos pontos P, A e B. Se $D_{ABP} > 0$, $D_{ABP} < 0$, ou $D_{ABP} = 0$, então o ponto P está do lado esquerdo, direito, ou sobre a linha de suporte AB, respectivamente. A Figura 4 ilustra uma representação gráfica de uma D-função para o caso $D_{ABP} > 0$.

1.2.3 Polígono *no-fit* (NFP)

O polígono *no-fit* tornou-se uma alternativa eficiente para lidar com a geometria de problemas de corte e empacotamento, uma vez que utiliza uma pesquisa iterativa e, igual ao caso da trigonometria direta, tem o benefício de representar as peças com maior precisão por usar as arestas originais de uma peça na sua representação. Um polígono *no-fit* é um polígono resultante de uma regra de interação de duas peças. A diferença vetorial da posição de dois polígonos pode ser usada para determinar se duas peças têm sobreposição, estão em contato ou se estão separadas, realizando um teste simples que

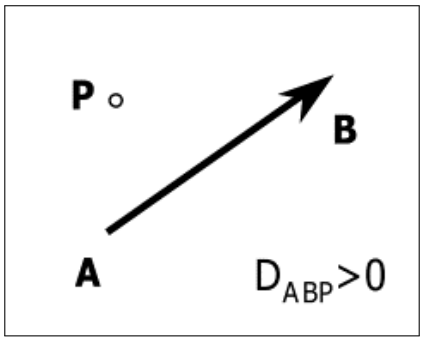


Figura 4: Interpretação da D-função. Fonte: Bennell & Oliveira (2008)

identifica se o vetor resultante está dentro do polígono *no-fit* (Bennell & Oliveira 2008).

O polígono *no-fit* pode ser resumidamente definido da seguinte maneira. Dados dois polígonos A e B que representam duas peças, o polígono *no-fit* de A e B é encontrado traçando uma forma ao redor do limite da outra, de modo que um dos polígonos tem que permanecer numa posição fixa, enquanto o outro percorre as bordas do polígono fixo, garantindo assim que os polígonos sempre estejam em contato, porém sem sobreposição (Burke et al. 2007). A Figura 5 ilustra a construção de um polígono *no-fit* considerando o quadrado A e o triângulo B. Note que A tem posição fixa na origem $(0, 0)$ e B é o polígono que se move ao redor do perímetro de A. O polígono *no-fit* é construído colocando B em uma posição de contato com A e marcando o local de um ponto de referência em B. O movimento ao redor é executado de forma que A e B sempre se tocam, porém sem sobreposição. O lugar geométrico do ponto de referência da peça B forma um caminho fechado que é o polígono *no-fit*, com isso, se o ponto de referência da peça B não estiver colocado dentro deste polígono, não existe sobreposição entre as peças, se está no contorno do polígono, as peças estão em contato mas sem sobreposição, e se está fora do NFP, as peças estão separadas.

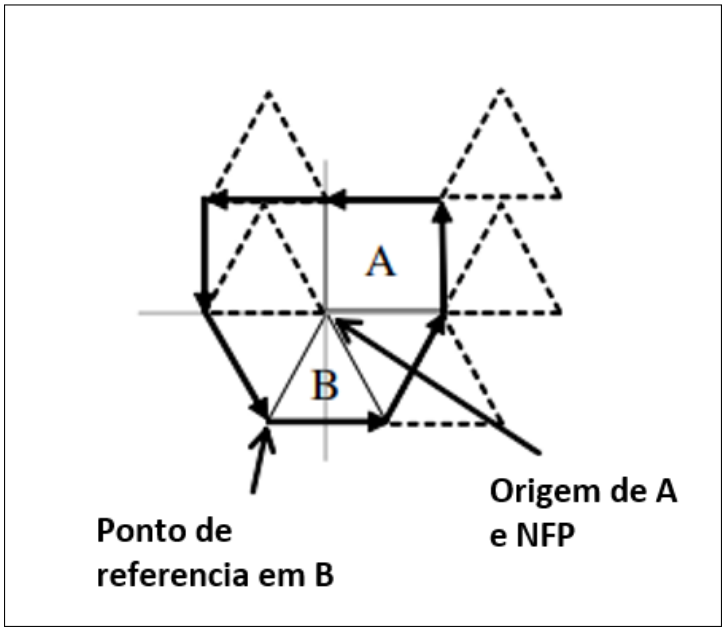


Figura 5: Traçando o movimento do polígono B em torno de A para formar o NFP. Fonte: Bennell & Oliveira (2008)

1.2.4 Função Phi

A função Phi é uma função matemática que representa as posições mutuas de duas peças, para o qual três casos podem ocorrer: o valor da função Phi é maior que 0 se as peças estiverem separadas, significando que não tem pontos em comum; o valor da função Phi é igual a 0 se seus limites estão em contato; e o valor da função Phi é menor que 0 se existe uma sobreposição entre elas, ou seja, pontos interiores em comum (Stoyan et al. 2002). Os valores da função devem ser uma medida da interseção das duas peças no terceiro caso e devem fornecer, ou pelo menos estimar, a distância entre as duas peças nos outros casos. O valor da distância entre os objetos é Euclidiana se a função Phi é normalizada, caso contrário é uma estimativa da distância. Seja $P(u)$ a representação de uma peça, onde u é o vetor de translação da peça. Sejam ainda $\text{int}P(u)$ e $\text{fr}P(u)$ o interior e a fronteira da peça, respectivamente. A função Phi estabelece uma regra de não sobreposição entre um par de peças $P_1(u_1)$ e $P_2(u_2)$, como segue.

Definição: Uma função contínua $f : \mathbb{R}^4 \to \mathbb{R}$ irrestrita é chamada de função ϕ de $P_1(u_1)$ e $P_2(u_2)$, denotada por $\phi_{12}(u_1, u_2)$, se atender as seguintes características:

$$\phi_{12}(u_1,u_2)\begin{cases} > 0, & \text{se } P_1(u_1)\cap P_2(u_2)=\emptyset, \\ = 0, & \text{se } \text{int}P_1(u_1)\cap \text{int}P_2(u_2)=\emptyset \text{ e } \text{fr}P_1(u_1)\cap \text{fr}P_2(u_2))\neq\emptyset, \\ < 0, & \text{se } \text{int}P_1(u_1)\cap \text{int}P_2(u_2)\neq\emptyset. \end{cases}$$

Seja γ o conjunto de pontos em $\mathbb{R}^4$ onde função ϕ_{12} é nula, ou seja, $\gamma := \{(u_1, u_2) \in \mathbb{R}^4 : \phi_{12}(u_1, u_2) = 0\}$. Este conjunto é chamado de superfície de nível 0 da função ϕ_{12}. Além disso, seja $G = \{(u_1, u_2) \in \mathbb{R}^4 : \text{int}P_1(u_1) \cap \text{int}P_2(u_2) = \emptyset\}$ o conjunto de posições factíveis (não sobrepostas) para $P_1(u_1)$ e $P_2(u_2)$ (Stoyan et al. 2004). Então, o conceito de função ϕ, para modelar problemas de posicionamento, permite trabalhar com casos onde as peças são não convexos e multiconectados, e até mesmo não conectados, assim como a região onde as peças devem ser alocadas pode ter essas características.

Adicionalmente, utilizando a definição acima para múltiplas peças é possível construir um modelo matemático para os problemas de empacotamento. Ou seja, se $\phi_{ij}(u_i, u_j)$ é uma função ϕ das i-ésima e j-ésima peças e, além disso, $\phi_i(u_i)$ uma função ϕ da i-ésima peça interagindo com o objeto (a peça precisa estar completamente contida no interior do objeto). Então, um modelo matemático para o problema de alocar n peças no interior de um objeto por meio da definição da função ϕ tem a seguinte forma. Encontrar o extremo de $F(X)$ sujeito a $X \in W$, onde W é o conjunto de restrições gerado pelo seguinte sistema de desigualdade, $\phi_{ij}(u_i, u_j) \geq 0$, $i < j = 1, 2, \ldots, n$, $\phi_i(u_i) \geq 0$, $i = 1, 2, \ldots, n$, e $X = (u_1, u_2, ..., u_n, h)$ é o vetor de variáveis, onde u_i é o vetor de translação da i-ésima peça e h representa as características métricas do objeto no qual as peças devem ser alocadas. Tal modelo geral permite adaptar métodos de otimização em modelos lineares e não lineares para obter soluções factíveis.

1.3 Técnicas de resolução e otimização

Para encontrar soluções factíveis para o problema de corte e empacotamento é possível partir de diferentes abordagens ou métodos de construção, que podem ser divididos em dois casos: os que utilizam soluções parciais, ou seja, é adicionada peça por peça até construir uma solução final; e os que utilizam soluções completas, onde é usada uma solução factível para realizar localmente pequenas alterações. Na sequência é detalhado a proposta de Bennell & Oliveira (2009) sobre como trabalhar com soluções parciais e soluções completas.

1. **Soluções parciais:** Trabalhar com soluções parciais significa que a solução é construída peça por peça. Esta abordagem pode produzir soluções de qualidade razoável com baixo custo computacional. A qualidade do procedimento de construção depende de duas características. A primeira está relacionada com a ordem sequencial de posicionamento das peças. Como as peças irregulares têm características distintas não padronizadas, decidir essa ordem pode ser uma tarefa difícil, isto porque as peças não fornecem informações sobre como poderiam encaixar melhor. Para considerar a ordem, normalmente escolhe-se entre randomização, ascendente, descendente ou ordenação de acordo com uma regra fixa, estabelecida geralmente de acordo com características, como área, comprimento, altura ou envoltório retangular, entre outras (Gomes & Oliveira 2002, Oliveira et al. 2000). A abordagem mais simples é a seleção aleatória, que também pode servir para gerar soluções iniciais em algoritmos que buscam soluções completas. A segunda característica está relacionada com a regra de posicionamento. Uma das regras mais conhecidas é *bottom left* (BL), que move de forma iterativa cada peça horizontalmente o mais para esquerda possível e depois desce de forma vertical até que seja capaz de repetir o movimento da peça de forma horizontal o mais à esquerda possível, ou até que encoste em outra peça ou na parte inferior do objeto. Essa regra também pode ser naturalmente aplicada em outras direções, o importante é que a posição final tem que estar dentro do objeto sem sobreposição com outras peças. A Figura 6 ilustra um exemplo da regra de posicionamento BL.

2. **Soluções completas:** Trabalhar com soluções completas significa realizar pequenas alterações sobre uma solução já construída. Uma rotina para obter uma solução completa precisa considerar alguns aspectos importantes para otimizar e manter a factibilidade no processo. Por exemplo,

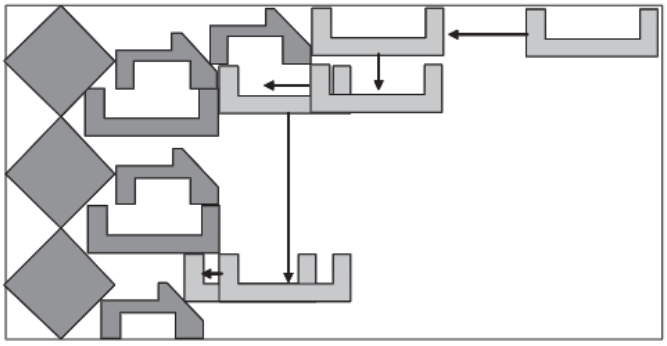


Figura 6: Regra de posicionamento *bottom left*. Fonte: Bennell & Oliveira (2009)

métodos que incorporam busca local devem considerar a função objetivo, o conjunto de soluções viáveis, o movimento de vizinhança e o mecanismo de busca dominante que controla a geração e aceitação de movimentos. Porém, a escolha da representação do problema é crucial para o desenvolvimento da estratégia de busca, para a qual existem duas representações principais do problema. A primeira envolve representar o problema como uma sequência de peças que é decodificada por meio de uma regra de posicionamento, e iterativamente troca a posição de peças dentro da sequência. A segunda envolve trabalhar diretamente com a solução, que pode ser chamada de layout físico, e aplicar movimentos das peças dentro do layout. A literatura apresenta algumas metaheurísticas para obter soluções completas para o problema, como a busca tabu, o recozimento simulado e algoritmos genéticos.

1.3.1 Metaheurísticas

Os problemas de corte e empacotamento são considerados *NP-hard*, o que justifica na literatura o uso de métodos heurísticos e metaheurísticos de resolução, sendo o recozimento simulado e algoritmos genéticos os mais usados. Na sequência esses métodos são exemplificados juntamente com detalhes de implementação para o problema de corte e empacotamento.

Recozimento simulado

O recozimento simulado é uma metaheurística baseada no processo físico pelo qual um sólido é resfriado lentamente para que, quando estiver congelado, a estrutura do material esteja em uma configuração de energia mínima. Esta técnica foi proposta inicialmente por Kirkpatrick et al. (1983) como estratégia para encontrar uma configuração de parâmetros que minimize alguma função objetivo. Esta função é uma medida da qualidade de uma configuração específica de parâmetros. O processo inicia com uma solução inicialmente factível e, para aperfeiçoá-la, perturbações são introduzidas na esperança de melhorar a solução. Explora-se então as configurações vizinhas mais acessíveis, escolhendo aquela que oferece um valor melhor conforme a função objetivo. Caso não se encontre uma configuração superior, recorre-se ao uso da variável temperatura, cujo valor influencia a decisão de aceitar ou não configurações inferiores.

A variável temperatura inicia com um valor alto, e diminui com cada iteração mediante um mecanismo de resfriamento até alcançar a temperatura final. O processo se repete a partir da nova configuração até que nenhuma melhoria adicional seja obtida. O método oferece uma estratégia semelhante à melhoria iterativa, com a diferença que o recozimento permite que, de forma controlada, as perturbações individuais alcancem posições de menor qualidade. Como cada movimento pode transformar uma configuração ou solução em uma pior, é possível sair dos mínimos locais e potencialmente cair num caminho descendente mais promissor (Rutenbar 1989).

Na literatura encontram-se aplicações do recozimento simulado para problemas de corte e empacotamento com boas soluções. Martins & Tsuzuki (2008) aplicaram o método para o problema de minimizar o desperdício de espaço que ocorre na alocação rotacional de um conjunto de polígonos bidimensionais irregulares dentro de um objeto. O conjunto de soluções viáveis consideradas pelos autores é o subconjunto de soluções conectadas. A Figura 7 ilustra um exemplo de uma solução viável e o seu

gráfico de conectividade, onde o grafo de conectividade relacionado a uma solução de um problema de posicionamento é o grafo cujos vértices são compostos pelo objeto P_0 e pelas peças P_i colocadas dentro dele. Utilizou-se o polígono *no-fit* para evitar sobreposição. Na proposta dos autores, para cada peça não alocada, realiza-se uma busca binária de profundidade limitada para encontrar um fator entre 0 e 1 que, quando aplicado à peça, permita ser encaixada no objeto. A colocação das peças é controlada por parâmetros ϕ relacionados com a rotação aplicada na peça e a posição relativa ao objeto e às peças já colocadas. Um parâmetro r define qual das muitas regiões possíveis da região livre de colisão pode colocar a peça. A Figura 8 ilustra um exemplo do procedimento para encontrar uma solução inicial e posteriormente aplicar o método de recozimento simulado.

Algoritmos genéticos

Os algoritmos genéticos são metaheurísticas baseadas na teoria da evolução, especificamente com o conceito da supervivência do mais apto, propostas inicialmente por John Holland em 1975, na Universidade de Michigan. Este tipo de método trabalha com população de soluções a cada iteração, onde são descartadas algumas soluções e substituídas por outras novas. Para o processo de substituição se tem duplas de soluções sobreviventes que transmitem algumas características às novas duplas de soluções, como se fossem proles originadas de pais. Essas características que se transmitem de geração a geração são chamadas de soluções factíveis que se organizam como cromossomos. Cada cromossomo é avaliado por uma função objetivo que permite diferenciar uma solução boa de uma ruim (Holland 1992). Para desenvolver um algoritmo genético deve-se partir de uma quantidade finita de cromossomos (população inicial) para tentar evoluir a população mediante regras probabilísticas guiadas pelos operadores descritos a seguir.

1. **Seleção:** operador utilizado para a seleção do cromossomo mais apto para engendrar uma nova geração. Distintos métodos podem ser aplicados para a seleção baseados em probabilidades.

2. **Cruzamento:** operador que realiza o cruzamento entre os cromossomos selecionados para gerar filhos, transmitindo aos cromossomos filhos suas melhores características para tentar melhorar a nova população. Na literatura é possível encontrar diferentes tipos de cruzamentos adaptados a diferentes problemas.

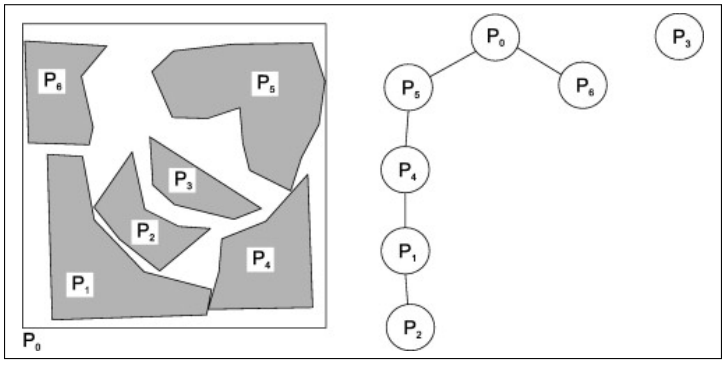


Figura 7: Solução viável e seu gráfico de conectividade. Fonte: Martins & Tsuzuki (2008)

Figura 8: Exemplo de posicionamento de uma peça durante o processo de otimização. Fonte: Martins & Tsuzuki (2008)

3. **Mutação:** operador que realiza alterações nos cromossomos. O objetivo é garantir que seja explorada a maior parte do espaço de soluções e evitar estar preso num ótimo local. O tipo de mutação dependerá da adaptação do método ao problema estudado e dos objetivos a serem alcançados.

4. **Avaliação:** operador que avalia o total de cromossomos (pais e filhos) mediante uma função objetivo que determinará quais soluções permanecem na população e quais são excluídas.

Existem variações interessantes na literatura baseadas nos algoritmos genéticos, como o algoritmo genético de chave aleatória viesada (*Biased Random Key Genetic Algorithm* - BRKGA), que difere do algoritmo clássico na configuração do cromossomo e na operação de cruzamento. O cromossomo é representado como um *array* de chaves aleatórias, nas quais cada chave é um número real entre 0 e 1. No cruzamento é obrigatório selecionar um cromossomo com características boas (elite) junto com qualquer outro, aumentando a probabilidade (superior ao 50%) de que as características boas do cromossomo elite sejam transmitidas ao novo cromossomo (Mundim et al. 2017).

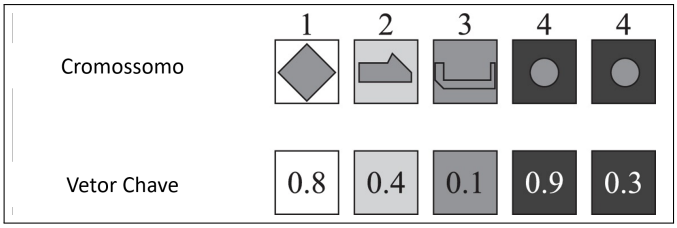
O BRKGA tem sido aplicado por diferentes autores para dar solução ao problema de corte e

empacotamento, mas como dito anteriormente, é preciso adaptar o algoritmo ao problema. Para cada cromossomo existe um vetor de n posições, em que n é o número total de peças disponíveis para empacotar e cada posição do vetor possui um número entre 0 e 1 (uma chave). A Figura 1.9(a) ilustra uma representação gráfica de um cromossomo associando peças e chaves. A orientação em que cada peça é colocada se obtém de sua chave no vetor. Para gerar novas soluções, utiliza-se o método *bottom left*, e para avaliar cada cromossomo considera-se o comprimento após empacotar as peças. A Figura 1.9(b) ilustra uma aplicação do BRKGA para o problema de corte e empacotamento.

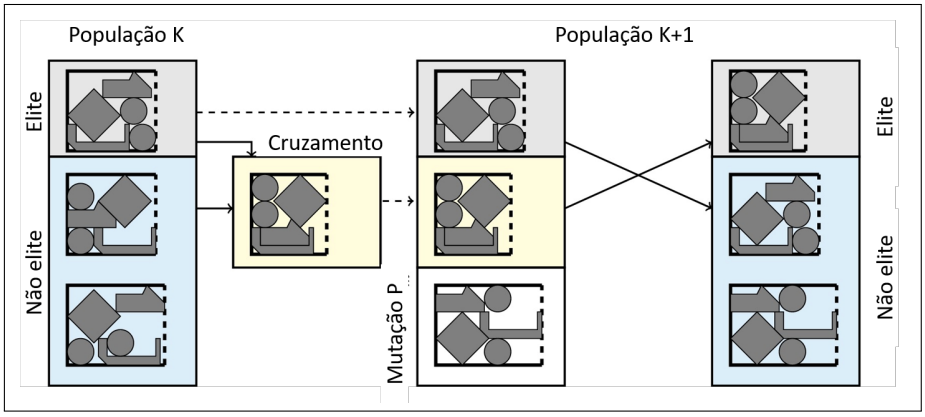
1.3.2 Aprendizado de máquina

O aprendizado de máquina é um campo em evolução de algoritmos computacionais que visam emular a inteligência humana. Ele vem se tornando cada vez mais importante na literatura devido a sua aplicação nos grandes volumes de dados disponíveis de variadas naturezas, como textos, imagens e áudios. Técnicas baseadas em aprendizado de máquina foram aplicadas com sucesso em diferentes áreas, como reconhecimento de padrões, visão computacional, engenharia de espaçonaves, finanças, entretenimento, biologia computacional e aplicações biomédicas e médicas. Em geral, as técnicas são baseadas em três escolhas: a estrutura, onde precisa-se do modelo que será utilizado para realizar a tarefa desejada, ou seja, está relacionada à caracterização do mapeamento entrada-saída gerado pelo modelo; o critério de adaptação, que expressa matematicamente o que se deseja atingir no processo de aprendizado, isto mostra o grau de sucesso alcançado pelo modelo; e um algoritmo de otimização responsável por encontrar os parâmetros ótimos do modelo.

O processo geral do aprendizado de máquina envolve utilizar na entrada uma base de dados para treinar o modelo com propósito de aprender e conseguir obter saídas corretas. Para isso, técnicas de



(a) Exemplo de cromossomo associando peças e chaves



(b) Processo do BRKGA para o problema nesting

Figura 9: BRKGA para o problema nesting. Fonte: Mundim et al. (2017)

validação são utilizadas para evitar o sobre-treinamento do modelo (El Naqa & Murphy 2015). O aprendizado de máquina possui os três seguintes paradigmas.

- **Supervisionado:** Para dado ou amostra de treinamento disponível, há uma resposta (saída) conhecida, ou seja, os dados são rotulados. Por causa disso, pode-se definir uma função custo que mostre uma medida de erro entre a saída que gerou o modelo e a esperada para cada dado do conjunto de treinamento.

- **Não-supervisionado:** Neste caso não se tem uma resposta associada a cada dado de treinamento, ou seja, os dados não são rotulados. Neste paradigma, espera-se que o modelo seja capaz de representar propriedades ou características do conjunto de dados. Em alguns casos é possível utilizar dados de treinamento rotulados e não rotulados, sendo assim um paradigma semi-supervisionado.

- **Reforço:** Neste caso, não se tem dados para treinar o modelo, porém se tem acesso a uma informação sobre a qualidade da resposta gerada pelo modelo na forma de um sinal de recompensa.

Para os problemas de corte e empacotamento, os métodos de aprendizado de máquina são comumente utilizados para apoiar a decisão de qual heurística ou meta-heurística terá melhor desempenho ao explorar informações sobre a instância atual do problema a ser resolvida, o qual também é chamado de hiper-heurística. As técnicas de aprendizado de máquina são implementadas com bons resultados como parte de uma metodologia na resolução de problemas de empacotamento, visando à sua otimização. Portanto, é importante continuar investigando seu potencial na resolução desse tipo de problema, desenvolvendo novas técnicas e aprimorando ou combinando as técnicas existentes (Gahm et al. 2022).

Q-learning

Q-learning é um algoritmo de aprendizado de máquina que pertence aos métodos de aprendizado por reforço, *Reinforcement Learning* (RL), em que um agente aprende a tomar decisões em um ambiente desconhecido por meio da própria interação com esse ambiente. Essas decisões são orientadas pela maximização de uma função chamada de função Q, representando o valor esperado acumulado de realizar uma ação em um determinado estado (Sutton & Barto 2018). A seguir estão descritos os principais componentes do *Q-learning*.

- **Espaço de Estados (S):** Representa todas as possíveis situações ou estados em que o agente pode se encontrar.

- **Espaço de Ações (A):** Representa todas as possíveis ações que o agente pode executar.

- **Função Q ($Q(s, a)$):** Indica o valor esperado acumulado da realização da ação "a" no estado "s". Essa função é atualizada iterativamente à medida que o agente explora o ambiente.

- **Recompensa (R):** representa a gratificação instantânea que o agente recebe ao executar uma ação em um determinado estado.

- **Taxa de Aprendizado (α):** Determina a rapidez com que o agente atualiza suas estimativas da função Q. Valores mais altos indicam uma adaptação mais rápida, mas podem levar a oscilações. Valores mais baixos tornam a aprendizagem mais estável, porém mais lenta.

- **Fator de Desconto (γ):** Representa a importância atribuída às recompensas futuras. Um valor próximo de 1 indica que o agente considera fortemente as recompensas futuras, enquanto que um valor próximo de 0 indica que o agente prioriza as recompensas imediatas.

A atualização da função Q ocorre através da equação de Bellman, que é uma equação de otimalidade. A fórmula de atualização é dada por:

$$Q(s,a) = (1-\alpha) * Q(s,a) + \alpha * (R + \gamma * max_a(s',a))$$

Onde:

- Q ($Q(s,a)$) é o valor Q para o estado s e a ação a.
- α é a taxa de aprendizado.
- R é a recompensa imediata
- γ é o fator de desconto.
- $max_a(s',a)$ é o valor Q máximo para o próximo estado s'.

Bartmeyer et al. (2022) desenvolveram uma técnica de aprendizado por reforço *Q-learning* incorporado em um sistema especializado para a resolução de problemas de empacotamento irregular, usado principalmente para superar a dificuldade da presença de defeitos no material durante um processo produtivo, que exige a geração de um novo layout em um tempo reduzido. Ficou concluído que esta técnica é adequada para gerar sequências de posicionamento, além de destacar que a sequência escolhida pode influenciar significativamente na qualidade das soluções obtidas nas diferentes regras de posicionamento consecutivo e, por esta razão, a sequência deve ser considerada um parâmetro a ser otimizado.

CAPÍTULO 2

Revisão da literatura

Uma breve revisão da literatura é apresentada com o objetivo de entender a evolução e o contexto atual relacionado com os problemas de corte e empacotamento, incluindo principalmente trabalhos que descrevem diferentes formas de abordar o problema de empacotamento com respeito à geometria e aos métodos de resolução exatos e heurísticos.

É possível encontrar na literatura tutoriais e revisões de abordagens e métodos de resolução para o problema de corte e empacotamento. Bennell & Oliveira (2008) forneceram um tutorial cobrindo as principais metodologias geométricas empregadas por pesquisadores no corte e empacotamento de peças irregulares. Os pontos fortes e fracos foram destacados pelos autores ao ilustrar a teoria e a implementação das abordagens, fornecendo uma guia sobre a implementação das técnicas mais conhecidas no momento. Leão et al. (2020) realizaram uma revisão dos modelos matemáticos propostos na literatura para problemas de corte e empacotamento, destacando as principais diferenças e semelhanças entre os modelos e suas vantagens e desvantagens para servir como ponto de inspiração para novas pesquisas na área.

Alguns métodos de resolução são desenhados ao considerar uma discretização uniforme da área do objeto e das peças, reduzindo a informação geométrica do objeto e das peças em uma representação em grade, que pode ser armazenada em uma matriz. Diferentes configurações produzem diferentes regras de alocação de peças que usam a informação geométrica reduzida. Baseado em uma representação em grade, Segenreich & Braga (1986) desenvolveram um algoritmo geral que utiliza a técnica de Monte Carlo para melhorar continuamente a solução, considerando as orientações das peças fixas para atingir resultados satisfatórios. Babu & Babu (2001) propuseram um método de representação das geometrias dos objetos e das peças de forma discreta, que permite encontrar soluções rapidamente por meio de uma heurística e um algoritmo genético, independentemente da complexidade geométrica das peças e dos objetos. Bennell & Oliveira (2009) descreveram e avaliaram as principais abordagens utilizadas até o momento para solucionar os problemas de corte e empacotamento de peças irregulares. As aplicações baseadas em soluções parciais e completas foram detalhadas pelos autores para destacar a necessidade do aprofundamento no tratamento eficiente da geometria. Baldacci et al. (2014) trabalharam com problemas de corte e empacotamento nos quais parte da superfície do objeto poderia ter defeitos que

deveriam ser evitados, porém para determinados tipos de peças alguns defeitos poderiam ser aceitos. Duas heurísticas e três procedimentos construtivos para alocação de peças para minimizar o desperdício total de material foram usados pelos autores na resolução do problema. Sato et al. (2023) propuseram algoritmos de separação e compactação que consideram a discretização da área do objeto e das peças, para resolver o problema de empacotamento bidimensional, com as duas dimensões abertas. Esses algoritmos realizam processos de compactação iterativos em que o passo chave é um algoritmo de separação baseado em um mapa de obstruções.

Abordagens de resolução exata também foram desenvolvidas por meio de formulações matemáticas que consideram discretização do posicionamento das peças na faixa, diferente das anteriores formulações em que o posicionamento é contínuo. Toledo et al. (2013) propuseram um modelo inteiro misto para o problema de empacotamento irregular que utiliza variáveis binárias associando cada ponto discreto de um tabuleiro (objeto) a cada tipo de peça. Para validar o modelo foram resolvidas na otimalidade uma quantidade significativa de instâncias dimensionalmente grandes. A qualidade das soluções obtidas pelo método dependem do grau de discretização e apresenta muito potencial para lidar com peças não convexas e objetos irregulares e/ou com defeitos. Leão et al. (2016) desenvolveram uma formulação inovadora de programação semi-contínua inteira mista para problemas bidimensionais de corte e empacotamento com peças de formato irregular. Os resultados computacionais mostraram que o modelo quando é resolvido por um solver comercial permite solucionar problemas com grande número de peças e determinar a solução ótima para instâncias dimensionalmente menores. Rodrigues & Toledo (2017) propuseram um modelo de cobertura de cliques baseado no modelo de quadro pontilhado proposto por Toledo et al. (2013). Os autores substituíram um grande número de restrições do modelo pontilhado por um conjunto menor, o que pode melhorar a relaxação linear. Além disso, propuseram um método para melhorar o limite inferior do modelo. Os resultados mostraram que o modelo proposto superou o modelo pontilhado original, uma vez que conseguiu resolver na otimalidade instâncias de 25 peças com até 22 peças distintas, enquanto o máximo no modelo anterior foi 21 peças com 7 peças distintas. Leão & Toledo (2021) implementaram uma estratégia de ramificação no método *branch-and-bound* e aplicaram um conjunto de desigualdades válidas ao modelo matemático semi-contínuo proposto por Leão et al. (2016). Com a nova formulação as autoras encontraram soluções ótimas em instâncias nas quais anteriormente não haviam sido reportadas.

Outros trabalhos utilizam os recursos da trigonometria direta na resolução deste tipo de problema e na construção de modelos matemáticos. As estratégias mais comuns são a função-D e o uso da representação de cobertura por círculos para descrever as peças. Rocha et al. (2014) apresentaram uma abordagem que auxilia na obtenção de layouts *nesting* altamente compactados com peças irregulares usando rotações livres. Propuseram três tipos de representação da cobertura por círculos para resolver o problema *nesting* no empacotamento de faixas por meio de um modelo de programação não linear. As representações usadas foram a cobertura de círculo completo, a cobertura do círculo interno e a cobertura parcial do círculo. Wang et al. (2018) estudaram o problema de corte e empacotamento para determinar a configuração de um conjunto de peças irregulares dentro de um objeto retangular de material com largura fixa, minimizando o comprimento do objeto. Os autores utilizaram polígonos em rotação livre e para aproximar a geometria das peças utilizaram um algoritmo chamado *QP-Nest* que

usa círculos inscritos nelas e assim dinamicamente melhora a qualidade da aproximação introduzindo novos círculos. Cherri et al. (2016) utilizaram modelos de programação matemática robustos para obter soluções do problema de empacotamento irregular com restrições de não sobreposição baseadas na trigonometria direta por meio da função-D. A abordagem utilizada mostrou-se robusta tanto em termos de estabilidade numérica quanto na variação dos tipos de geometria, considerando qualquer tipo de peça não convexa e qualquer tipo de furo em seu interior. Kimms & Király (2023) propuseram um modelo linear inteiro misto para o problema de empacotamento de peças irregulares em faixas, que implementa trigonometria direta na verificação de sobreposição entre peças. Para diminuir a complexidade da verificação de sobreposição, eles implementaram a estratégia de, entre duas peças, definir para cada aresta de uma delas, um vértice crítico da outra, com o qual realiza uma única verificação.

Tem sido amplamente estudado o efeito da sequência de posicionamento das peças nas regras de posicionamento consecutivo, como o caso das heurísticas BL. Estas regras de ordenação são estabelecidas geralmente com base em critérios relacionados à geometria das peças. Romanelli (2013) estudou o efeito de diferentes regras de ordenação em seu algoritmo de resolução proposto, no qual os atributos observados foram área do polígono, perímetro externo, comprimento, largura, concavidade e não-retangularidade, ordenados de maior valor a menor. Além dos atributos anteriores, utilizou um critério adicional denominado valor agregado, calculado pela soma das posições obtidas por cada peça nas ordenações feitas com os outros atributos observados. A ordenação das peças pelo valor agregado é feita no sentido do menor valor para o maior. Neste mesmo trabalho foi proposta uma estratégia de compactação para o layout parcialmente construído, que alterna etapas de posicionamento de peças com etapas de compactação. Qi et al. (2023) propuseram um algoritmo híbrido de técnicas de otimização para o empacotamento de peças irregulares e implementaram uma estratégia robusta de ordenação da sequência de posicionamento das peças, baseada no algoritmo de predador marinho (MPA).

Metaheurísticas são frequentemente utilizadas na literatura para resolver problemas de corte e empacotamento. Oliveira & Ferreira (1993) apresentaram um algoritmo de layout automático para o problema de corte e empacotamento baseado na abordagem de recozimento simulado. O algoritmo foi implementado em dois formatos, para os quais o primeiro assume que as peças irregulares são representadas em forma discreta e o segundo assume que as peças são representadas por um conjunto de vértices que conectados por segmentos definem um polígono. Martins & Tsuzuki (2008) trabalharam com o problema de minimização de desperdício de espaço na colocação rotacional de um conjunto de peças bidimensionais irregulares dentro de um objeto bidimensional. Os autores aplicaram a técnica de verificação de não sobreposição NFP e, para obter solução do problema, foi utilizado o método de recozimento simulado. Domovic et al. (2014) aplicaram a um conjunto de peças regulares e irregulares, métodos de busca aleatória, algoritmos gulosos e genéticos, visando encontrar o melhor posicionamento das peças que minimize a área de cobertura. Os autores usaram NFP para garantir que não haja sobreposição entre as peças. Mundim et al. (2017) aplicaram um algoritmo genético de chave aleatória para definir a rotação e sequência em que cada peça é empacotada no problema de corte e empacotamento irregular, incluindo um procedimento *no-fit raster* para a verificação de sobreposição. As instâncias testadas para validar o método foram de médio e grande porte.

Diyaley & Chakraborty (2022) apresentaram seis metaheurísticas para solucionar o problema de corte e empacotamento de peças irregulares em uma chapa de metal com limite restrito, de modo a maximizar a utilização efetiva do material com o mínimo de desperdício. As técnicas aplicadas foram colônia de abelhas, colônia de formigas, enxame de partículas, algoritmo *firefly*, evolução diferencial e algoritmo baseado em ensino-aprendizagem. Todos os métodos conseguiram encontrar boas soluções, sendo o último quem conseguiu os melhores resultados. Bartmeyer et al. (2022) desenvolveram uma abordagem que utiliza a técnica de aprendizado de máquina chamada aprendizado por reforço para gerar sequências de alocações otimizadas para o problema de corte e empacotamento. A abordagem consiste da integração da heurística de posicionamento *bottom left*, da técnica de verificação de sobreposição NFP e do aprendizado por reforço, para obter um sistema especialista que consegue se ajustar às variações nas características do material, como o surgimento de defeitos no objeto no processo de corte, gerando um novo layout otimizado em poucos segundos. Liu et al. (2023) propuseram um algoritmo híbrido chamado GA-LP, para resolver o problema de empacotamento irregular bidimensional, visando aproveitar conjuntamente a capacidade de busca global de um algoritmo genético com as características da resolução exata da programação linear. A técnica alcança uma solução inicial com o algoritmo genético e posteriormente otimiza essa solução por meio da programação linear.

CAPÍTULO 3

Descrição do problema e método de resolução proposto

Este capítulo apresenta a descrição do problema de empacotamento irregular que foi resolvido neste trabalho e o método proposto para sua resolução. Utilizamos a tipologia de Wäscher et al. (2007) para descrever o problema estudado. Adicionalmente, abordamos quatro componentes principais na construção da heurística de interação gravitacional proposta como método de resolução: i) a representação das peças que define geometricamente o conjunto de peças a serem posicionadas e a estrutura computacional implementada para resolver o problema; ii) a técnica de verificação de sobreposição; iii) a regra de posicionamento; e iv) o componente de otimização baseado em diferentes técnicas e estratégias presentes na literatura e em hipóteses propostas a partir das características específicas da heurística.

3.1 Descrição do problema

- **Dimensionalidade:** bidimensional.
- **Tipo de atribuição:** minimização de entradas.
- **Variedade das peças:** peças idênticas, variedade fracamente heterogêneo e variedade fortemente heterogêneo.
- **Variedade dos objetos:** um objeto grande de formato retangular com uma dimensão fixa e outra variável.
- **Formato das peças:** irregular.

A tipologia considera que itens idênticos com orientação diferente são tratados como itens diferentes. Porém, em relação à orientação dos itens, existem possibilidades diferentes conforme mencionado na revisão da literatura e vale especificar que o tipo de problema estudado contempla orientações discretas com incrementos de 1°.

3.2 Método de resolução

Na literatura existem técnicas de resolução para o problema de empacotamento de peças em faixas, baseadas no comportamento dos corpos de acordo com leis físicas, como é o caso do trabalho de Liao et al. (2016) que tenta simular o comportamento de peças irregulares em presença de um elástico que tende a juntá-las e compactá-las, com o qual alcançaram bons resultados tanto na qualidade das soluções como na otimização do custo computacional. O método proposto neste trabalho também busca simular o comportamento de corpos seguindo (respeitando) leis físicas, mas desta vez inspirado na interação gravitacional, por meio da integração de uma regra de posicionamento consecutivo de peças, que tenta simular o comportamento de corpos que giram sobre si (em torno do seu próprio eixo) e se transladam em torno de outros, com uma técnica de verificação de sobreposição baseada no cálculo da distância de colisão entre duas peças (estruturas geométricas).

O método visa reduzir significativamente a verificação de infactibilidades e soluções factíveis sem potencial de melhorias por meio de uma regra de posicionamento consecutivo de peças e da validação de hipóteses que restringem o conjunto de configurações. Adicionalmente, melhorias no desempenho do método são obtidas ao adicionar uma regra de ordenação, uma estratégia de compactação, uma metaheurística de recozimento simulado e escolhas dinâmicas de orientações permitidas. No fluxograma da Figura 10 são apresentadas de forma geral as diferentes etapas do método de resolução proposto, onde sequencialmente são feitas a leitura do problema, a representação das peças, a construção da solução inicial e a otimização da solução.

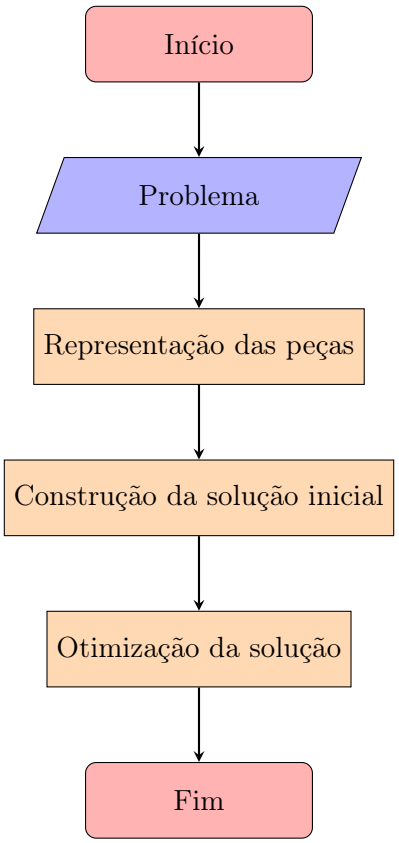


Figura 10: Fluxograma geral do método de resolução proposto. Fonte: Própria

3.2.1 Representação das peças

A representação das peças utiliza a trigonometria direta. Ela permite trabalhar com a geometria verdadeira das peças e do objeto porém contendo um custo computacional significativo. No entanto, para diminuir este custo computacional, foi proposta uma estratégia de armazenamento de parte das informações de verificação de sobreposição. Então, o método ganha tempo ao aproveitar esse armazenamento em futuras revisões de sobreposições.

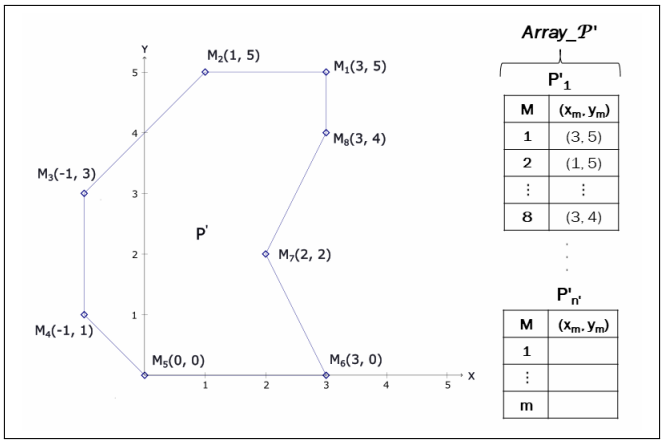


Figura 11: Representação de peças por arranjo de coordenadas. Fonte: Própria

Seja $\mathcal{P}' = \{P'_1, P'_2, \ldots, P'_{n'}\}$ um conjunto de polígonos simples representando a geometria das peças, tal que cada elemento é apenas uma cópia de cada tipo de peça do problema. Esta escolha reduz significativamente a quantidade de informações na representação das peças e tem potencial de reduzir o custo computacional. De acordo com a notação apresentada, a lista de vértices $(x_1, y_1), \ldots, (x_m, y_m)$ é armazenada em um arranjo de coordenadas ordenadas (x, y) denominado $Array_\mathcal{P}'$, conforme ilustrado na Figura 11, a qual também destaca o polígono P'_1. Posteriormente, foi realizada a translação de cada um dos polígonos do conjunto $\mathcal{P}'$, movendo o centroide de cada polígono para a origem do plano cartesiano e estabelecendo o centroide como o ponto de referência da peça.

A Figura 12 ilustra o polígono apresentado na Figura 11 após a aplicação da função de translação. Tanto para a etapa de construção da solução inicial, quanto para a etapa de otimização, são necessárias uma técnica de verificação de sobreposição das peças e uma regra de posicionamento, que serão detalhadas na próxima seção. No entanto, para sua implementação, é necessário incluir alguns parâmetros na representação dos polígonos, que são obtidos a partir das equações da reta de cada uma das suas arestas. Conforme ilustrado na Figura 12, cada aresta é representada pelos parâmetros w_0 e w_1, que são os coeficientes da equação da reta no formato $Y = w_0 + w_1 X$. O coeficiente w_0 (termo constante) e o inverso do coeficiente w_1 (coeficiente angular), são armazenados no arranjo $Array_\mathcal{P}'$ para cada orientação possível de uma peça.

3.2.2 Construção da solução inicial e otimização da solução

A distância mínima para que não exista sobreposição entre peças é utilizada para acomodar uma nova peça no layout e para realizar a verificação de sobreposição com as peças já posicionadas. Esta distância também é denominada de distância de colisão, que é usada em uma regra de posicionamento

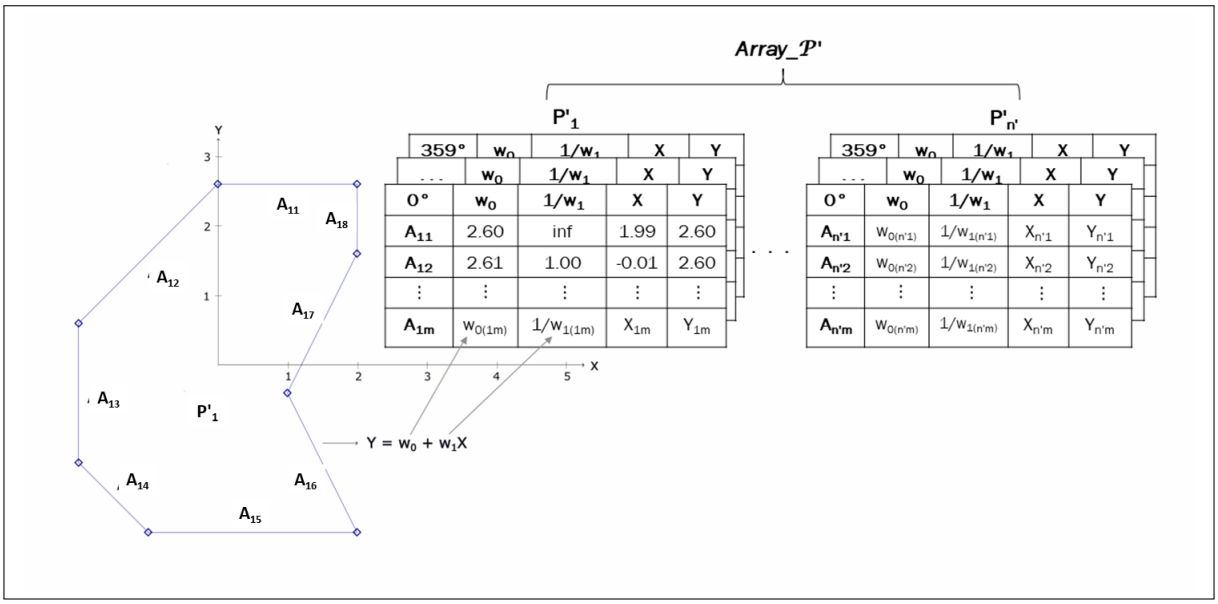


Figura 12: Representação de peças por arranjo de parâmetros de linha reta das arestas e coordenadas dos vértices. Fonte: Própria

consecutivo de peças, tanto na etapa de construção da solução inicial, quanto na etapa de otimização da solução inicial.

Verificação de sobreposição de peças

Como apresentado no Capítulo 2, as soluções para esse problema devem satisfazer restrições que garantam que as peças não apresentem sobreposições. Existem diferentes técnicas para verificar a

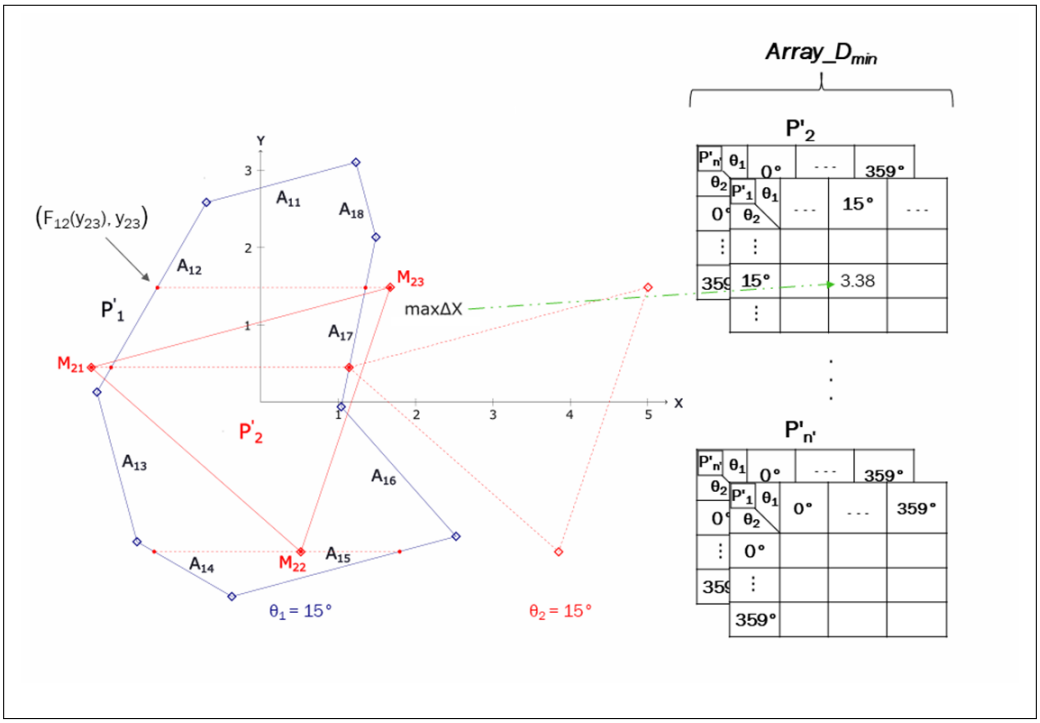


Figura 13: Representação de peças por arranjo de parâmetros de reta e coordenadas. Fonte: Própria

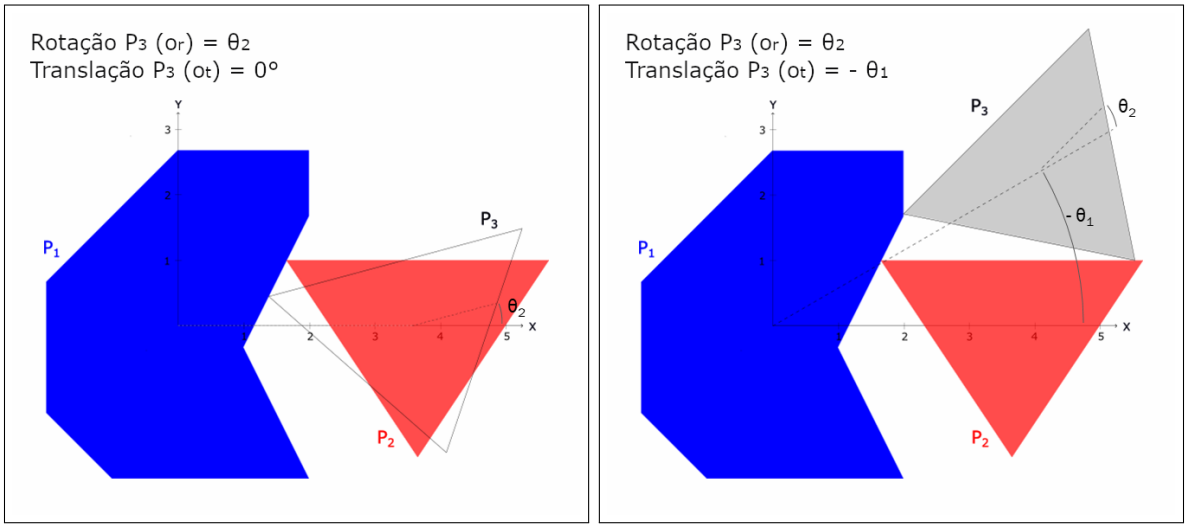
sobreposição entre as peças, tais como *no-fit polygon*, *no-fit raster*, funções phi, retas separadoras e técnicas baseadas em triângulos, entre outras. Para este trabalho foi proposta uma técnica de verificação de sobreposição baseada no cálculo da distância mínima permitida entre duas peças, dado um ângulo de rotação para cada uma delas. Essa distância mínima corresponde à distância de colisão entre as peças nas suas rotações específicas, conforme mostrado no exemplo na Figura 13, onde os ângulos de rotação da Peça 1 e da Peça 2 são θ_1 e θ_2 respectivamente. Essa técnica foi escolhida por dois motivos principais. O primeiro é que técnicas como *no-fit polygon*, *no-fit raster* e funções phi apresentam dificuldades quando o problema permite rotações livres (Nunes 2021). O segundo motivo, de grande importância, é que as verificações de sobreposição realizadas podem ser aproveitadas em verificações posteriores, otimizando o custo computacional. Isso é alcançado armazenando num arranjo as distâncias de colisão calculadas para posteriormente serem simplesmente lidas quando for necessário.

Na seção anterior, a representação das peças foi ampliada para incluir os parâmetros das equações de linha reta das arestas de cada polígono. A Figura 13 mostra como calcular a distância mínima entre a peça a ser posicionada j (P_2') e a peça já posicionada i (P_1'). Por meio da função $F_{im}(y_{jm'}) = (y_{jm'} - w_{0(im)})/w_{1(im)}$ que fornece a coordenada x do ponto em que o vértice m' da peça j colidirá com a aresta m da peça i, para um deslocamento somente na direção x. Esse cálculo deve ser realizado somente para pares vértice-aresta que tenham uma opção de colisão. Os valores Δx, correspondentes a $F_{im}(y_{jm'}) - x_{jm'}$, são armazenados em uma lista. Posteriormente, o mesmo procedimento é realizado para os vértices da peça i nas arestas da peça j, mas o Δx nesse caso corresponde a $x_{im} - F_{jm'}(y_{im})$. Esses valores Δx são incluídos na mesma lista para obter o máximo Δx, que corresponderá à distância de colisão e também à distância mínima permitida entre as duas peças para essas rotações específicas, que é armazenada no arranjo $Array_D_{min}$.

Regra de posicionamento

Neste trabalho foi proposta uma regra de posicionamento consecutiva inspirada na interação de corpos pela atração gravitacional, que chamaremos simplesmente de interação gravitacional (IG). Nessa abordagem, a posição da primeira peça do conjunto de peças a ser posicionado é definida no ponto de origem das coordenadas, com uma rotação específica. O posicionamento de cada uma das peças subsequentes, deve-se realizar em relação a uma peça de referência (P_r) dentro do conjunto de peças já posicionadas. Serão estabelecidos para o posicionamento da peça, um ângulo de translação (o_t) com base na peça P_r, e outro de rotação (o_r) em relação ao seu próprio ponto de referência (centroide neste caso), conforme mostrado na Figura 14. Esses ângulos são respectivamente $-\theta_1$ e θ_2, apresentados nesta subseção, com os quais é possível obter a distância de colisão entre as peças. O ângulo de translação está relacionado a $-\theta_1$ porque o cálculo da distância de colisão é realizado com base em suas rotações específicas, e transladar a Peça 2 em relação à Peça 1 equivale a girar a Peça 1 no sentido oposto. Com a distância de colisão entre as peças, o ângulo de translação e o ângulo de rotação, estão disponíveis todas as informações relativas entre a peça a ser posicionada e sua peça de referência, necessárias para o posicionamento.

Nessa primeira implementação da regra de posicionamento IG, a peça a ser posicionada (P_p), sua peça de referência (P_r), o ângulo de rotação da peça a ser posicionada (o_r) e o ângulo de translação



(a) Posicionamento inicial da Peça 3 com rotação θ_2 e translação 0°, em relação à Peça 1.

(b) Posicionamento final da Peça 3 com rotação θ_2 e translação $-\theta_1$, em relação à Peça 1.

Figura 14: Exemplo da regra de posicionamento IG. Fonte: Própria

em relação à peça de referência (o_t), são selecionados de forma consecutiva, a fim de observar o efeito da inclusão das diferentes estratégias de otimização. A Figura 3.14(a) mostra a primeira etapa da implementação dessa regra, que consiste em estabelecer a peça a ser posicionada, nesse caso a Peça 3, e depois percorrer o conjunto de peças posicionadas de forma ordenada para estabelecer a peça de referência, nesse exemplo a Peça 1 como a primeira alternativa. Em seguida, o ângulo de rotação o_r da peça a ser posicionada deve ser definido. No exemplo atual, a Peça 3 tem uma rotação θ_2. Por fim, o ângulo de translação $o_t = -\theta_1$ com relação à peça de referência (Peça 1) deve ser definido, que no exemplo é 0° inicialmente.

A próxima etapa é verificar se as restrições de sobreposição e que todas as peças estão completamente contidas no objeto, foram atendidas. Primeiramente, essa segunda restrição é verificada, levando-se em conta que ela exige menos recursos computacionais do que a primeira. Ela consiste em uma verificação simples que é realizada dinamicamente com o posicionamento de cada peça, atualizando a menor e a maior coordenada x, que é a dimensão do objeto estabelecida como fixa, e atualizando a menor e a maior coordenada y, que corresponde à dimensão variável do objeto ou ao comprimento. A atualização anterior é temporária, para todo o conjunto de peças já posicionadas, acrescentando a peça a ser posicionada, pois se alguma das restrições não for atendida, a peça não será posicionada e, portanto, as coordenadas máxima e mínima em cada dimensão correspondem àquelas atualizadas no posicionamento da última peça. A partir dessas coordenadas, o delta em cada uma das dimensões é calculado e comparado com os respectivos limites permitidos. No caso da dimensão x, ele é comparado com o valor fixo definido para essa dimensão e, no caso da dimensão y, ele é comparado com o comprimento da melhor solução encontrada até o momento, ou seja, com o comprimento da solução atual. Para a construção da primeira solução, o limite do comprimento é igual a infinito. Se tanto o Δx quanto o Δy forem menores ou iguais a seus respectivos limites, então todas as peças estão completamente contidas no objeto.

Para a verificação de sobreposição, é calculada a distância euclidiana entre a peça a ser posicionada

e a peça a ser verificada (cada uma das peças já posicionadas exceto a peça de referência). O arranjo $Array_D_{min}$ é revisado para ver se contém a distância mínima permitida entre as duas peças, para essas rotações específicas; se não existir, ela será calculada e armazenada no arranjo. As duas distâncias são então comparadas e, se a distância euclidiana calculada entre as duas peças for maior ou igual à distância mínima permitida entre as duas peças, então as peças não se sobrepõem. Essa estratégia de calcular a distância mínima e armazená-la em um arranjo, otimiza computacionalmente a verificação, pois uma configuração de rotações específicas é calculada apenas uma vez e só precisa ser lida do arranjo em comparações futuras.

Se uma das duas restrições acima não for atendida, o ângulo de translação o_t é aumentado em 1° ou em um valor de passo definido (pode ser qualquer valor entre 1° e 359°); se seu valor máximo (359°) for atingido, ele é definido como 0° e o índice da peça definida como peça de referência será aumentado em 1, para tentar novamente com incrementos no ângulo de translação. Se o valor máximo do índice da peça de referência for atingido, a Peça 1 é definida como a peça de referência e o ângulo de rotação o_r é aumentado em 1° ou em um valor de passo definido. Esse processo é executado até que um posicionamento factível seja encontrado ou até que o limite de ângulos de rotação seja atingido, quando então se estabelece que não há posicionamento factível para a peça atual, e o índice da peça a ser posicionada deve ser diminuído em 1, continuando a verificação do posicionamento factível, a partir do posicionamento atual da nova peça. A Figura 3.14(b) mostra o posicionamento fatível alcançado pela Peça 3 seguindo a sequência descrita, onde se estabelece como peça de referência a Peça 1, depois se estabelece o ângulo de rotação $o_r = \theta_2$ e posteriormente se translada desde o ângulo $o_t = 0°$ até o ângulo $o_t = -\theta_2$, onde finalmente o posicionamento atende as restrições.

O Algoritmo 1 corresponde ao pseudocódigo da regra de posicionamento IG proposta, que como pode ser visto, recebe como parâmetros o conjunto $\mathcal{P}'$ de peças (uma cópia de cada geometria) e o conjunto de índices denominado $ind(\mathcal{P})$ necessário para relacionar os elementos do conjunto total de peças a serem posicionadas $\mathcal{P}$ com o tipo de peça do conjunto $\mathcal{P}'$, de modo que o conjunto $ind(\mathcal{P})$ permite estabelecer a sequência de posicionamento das peças do conjunto $\mathcal{P}$ e relacioná-las com o seu tipo de peça. Além disso, recebe também a dimensão fixa do objeto, que, nesse caso, corresponde ao Δx máximo permitido. A saída do algoritmo corresponde aos vetores de translações v e rotações o, que são aplicados ao conjunto $\mathcal{P}$ para obter uma solução.

O algoritmo base é simplesmente um conjunto de laços aninhados que percorrem todas as combinações de posicionamento possíveis para uma sequência de peças. Internamente, cada peça da sequência $ind(\mathcal{P})$ tenta se posicionar em relação a outra peça já posicionada, onde a sequência de laços aninhados permite percorrer os ângulos de translação entre a peça P_p a ser posicionada e a peça de referência P_r, os ângulos de rotação da peça a ser posicionada e as peças de referência no conjunto de peças já posicionadas. Se a viabilidade for alcançada, a peça é posicionada, e se a solução for melhorada ao completar o posicionamento de todas as peças, a melhor solução é atualizada.

Para percorrer o conjunto de soluções de forma ordenada, é utilizado o arranjo $Array_IG$, que, durante a etapa de construção das soluções permite identificar, para a peça cuja posição tentaremos modificar, sua peça de referência P_r, seu ângulo de translação em relação à peça de referência o_t e seu ângulo de rotação em relação à seu próprio eixo o_r, conforme mostrado na linha 8 do Algoritmo 1.

Algoritmo 1: IG

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}
Saída: v, o

```
1  Array_P' ← Representação das peças;
2  Array_D_min ← null ou Matriz_D_min;
3  Array_IG ← null;
4  ΔY_max ← inf;
5  enquanto critério de parada não atingido faça
       /* Solução inicial e otimização                                  */
6      enquanto (o_r1 em O_r) e (critério de parada não atingido) faça
7          enquanto (P_p em ind(P)) e (P_p > 1) faça
8              {o_r, P_r, o_t} ← Array_IG;
9              enquanto (o_r em O_r) e ((não factível) ou (ΔY ≥ ΔY_max)) faça
10                 enquanto (P_r < P_p) e ((não factível) ou (ΔY ≥ ΔY_max)) faça
11                     enquanto (o_t em O_t) e ((não factível) ou (ΔY ≥ ΔY_max)) faça
12                         Verificação de restrições;
13                         se (factível) e (ΔY < ΔY_max) então
14                             Array_IG ← {o_r, P_r, o_t};
15                         o_t ← o_t + 1;
16                     P_r ← P_r + 1;
17                 o_r ← o_r + 1;
18             se ((não factível) ou (ΔY ≥ ΔY_max)) então
19                 P_p ← P_p − 1;
20         se (factível) e (ΔY < ΔY_max) então
21             ΔY_max, v, o ← Função(Array_IG);
22         o_r1 ← o_r1 + 1;
       /* Reordenar ind(P)                                              */
23     ind(P) ← Reordenar(ind(P));
```

Durante o posicionamento de cada peça, o arranjo *Array_IG* é atualizado e recebe P_r, o_t e o_r conforme mostrado na linha 14 do Algoritmo 1. Tanto no estágio de construção da solução inicial quanto no estágio de otimização, o comprimento máximo ΔY_{max}, o vetor de translações v e o vetor de rotações o são obtidos por meio de uma função que recebe o arranjo *Array_IG*, conforme mostrado na linha 21 do Algoritmo 1.

Outra característica marcante dessa regra de posicionamento é que ela consegue trabalhar com polígonos não convexos, permitindo o ajuste entre as peças, sem exigir qualquer tratamento prévio dos polígonos, como é estritamente necessário para algumas regras. Além disso, as mesmas técnicas implementadas em outras regras podem ser aplicadas para representar um polígono não convexo por meio de dois ou mais polígonos convexos, para casos de geometrias mais complexas, adaptando a regra de posicionamento para essa finalidade. Um critério de parada deve ser definido para delimitar a execução do algoritmo, e este pode ser parametrizado por número de iterações, limite de soluções ou limite de tempo.

Para restringir o conjunto de soluções a serem avaliadas foram propostas três hipóteses: a primeira para evitar movimentos sem opção de melhoria na qualidade da solução, a segunda, para usar dicioná-

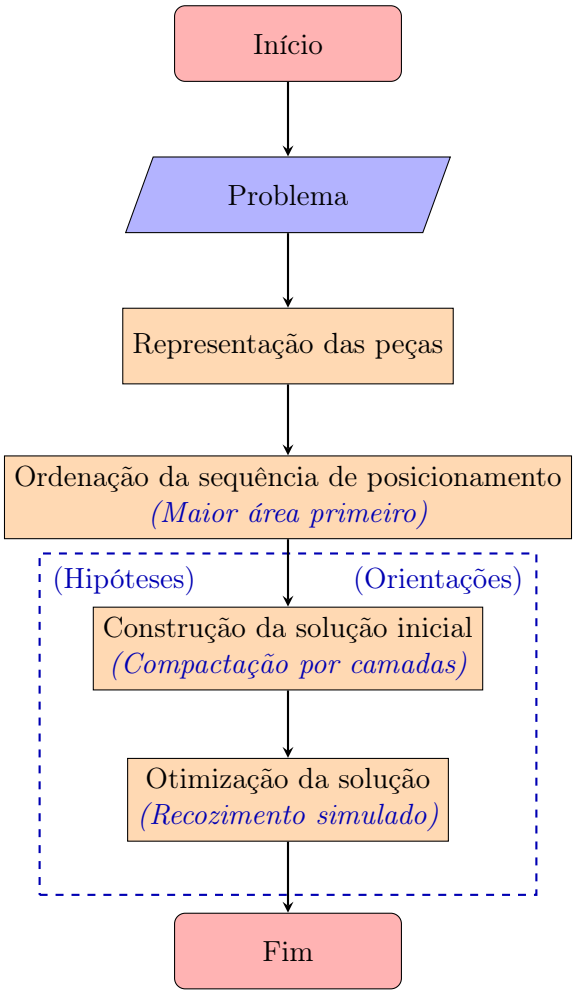


Figura 15: Fluxograma do método de resolução proposto incluindo as estratégias de otimização. Fonte: Própria

rios como memória de armazenamento de regiões infactíveis, e a terceira, para verificar sobreposição em ordem de maior a menor chance de ocorrência. Como pré-processamento foi aplicada a regra de ordenação de maior área primeiro, para a escolha da sequência de posicionamento das peças. Na etapa de construção da solução inicial foi implementada uma estratégia de compactação por camadas e na etapa de otimização foi implementada uma metaheurística de recozimento simulado. Para melhorar o desempenho computacional do método, foi proposta uma estratégia de escolha dinâmica de orientações permitidas (orientações).

O fluxograma apresentado na Figura 15 descreve detalhadamente o método proposto para resolver o problema, destacando em azul as diversas técnicas e estratégias implementadas para otimizar o desempenho do método. Nele, a regra de ordenação por maior área primeiro é aplicada inicialmente como um pré-processamento após a representação das peças. As hipóteses formuladas e a estratégia de escolha dinâmica das orientações permitidas estão integradas tanto no processo de construção da solução inicial quanto no processo de otimização ou compactação. A estratégia de compactação por camadas é empregada exclusivamente durante a fase de construção da solução inicial. Em seguida, a metaheurística de recozimento simulado é implementada na fase de otimização da solução para melhorar ainda mais a solução.

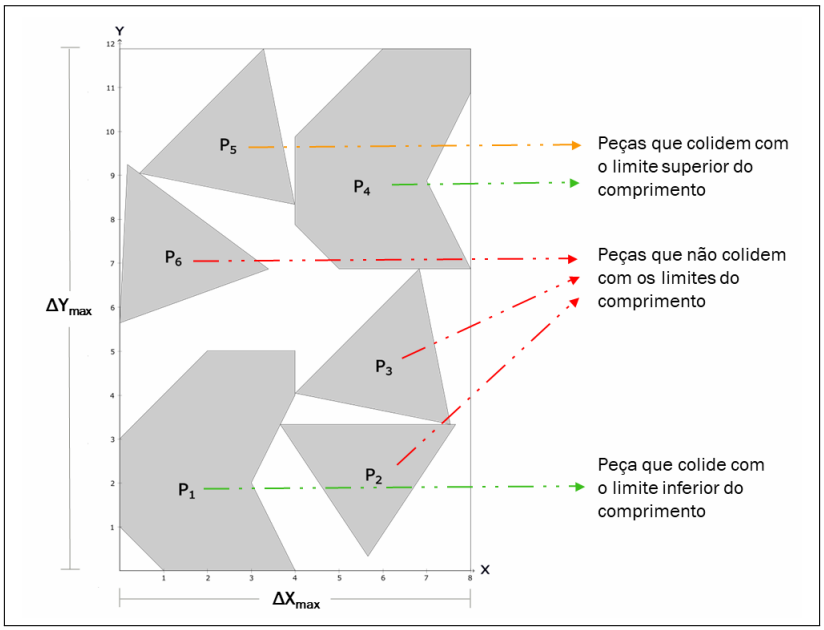


Figura 16: Classificação das peças em relação aos limites do comprimento. Fonte: Própria

3.2.3 Otimização da regra por hipóteses

Movimentos com opção de otimização

A mudança de posição de uma peça tem a possibilidade de melhorar a solução, se e somente se, para um dos limites do comprimento, todas as peças que com ele colidirem tiverem um índice maior ou igual ao da peça. Na regra de posicionamento consecutivo proposta, a relação das peças com os limites e os seus índices permite-nos determinar se a modificação da posição de uma peça tem o potencial de otimizar a solução.

A Figura 16 mostra um exemplo de posicionamento de 6 peças. No exemplo, o deslocamento da Peça 6 não tem potencial de otimização porque os limites do comprimento permanecerão os mesmos, uma vez que nenhuma outra peça será modificada, de acordo com a sequência. A Peça 5 está em contacto com um dos limites do comprimento, no entanto, mudar a posição desta peça e depois posicionar a Peça 6 também não tem qualquer potencial de melhoria, porque o comprimento permanecerá o mesmo, uma vez que a Peça 4 no limite superior e a Peça 1 no limite inferior não seriam modificadas. Agora, movimentar a Peça 4 tem sim potencial de melhoria, pois nenhuma peça com índice inferior a esta, está no limite superior, o que significa que este limite pode mudar quando a sequência de peças for reposicionada a partir da Peça 4.

Otimização com uso de dicionários como memória

Na linguagem de programação *Python* os dicionários correspondem a uma estrutura de dados, também conhecida como "*memória associativa*". Através desta estrutura é possível adicionar ao algoritmo um componente de memória, que permite armazenar os intervalos de posicionamento infactíveis dentro do conjunto de soluções, evitando repetir a revisão de configurações já processadas.

Para melhor compreender esta hipótese, no exemplo da Figura 3.14(b), observa-se que antes de

poder posicionar a Peça 3, o conjunto de soluções foi percorrido de $o_t = 0$ até $o_t = -\theta_1 - 1$, o que corresponde a um intervalo de posicionamento infactível da Peça 3 dado o posicionamento das demais peças. Este intervalo é armazenado no dicionário para evitar que a mesma peça ou outra peça do tipo 3, ou seja, com a mesma geometria, percorra este intervalo em uma configuração de peças iguais. Para otimizar o cálculo da atualização do dicionário, ele é realizado uma vez posicionada a peça, realizando uma única atualização para todo o intervalo.

Verificação de sobreposição em ordem da chance de ocorrência

Foi formulada a hipótese de que o desempenho computacional pode melhorar ao verificar a restrição de sobreposição por ordem da chance de ocorrência, começando pela peça com maior chance de sobreposição e seguindo em ordem decrescente, reduzindo assim o número de verificações necessárias. Para isso, em uma lista que representa a chance de sobreposição da peça a ser posicionada com cada uma das peças já alocadas, as maiores chances de sobreposição estão inicialmente associadas às peças que compartilham a mesma peça de referência que a peça a ser posicionada. Adicionalmente, a cada revisão, caso haja sobreposição com a peça verificada, o respectivo aumento nesta chance é atualizado. Com o exposto, antes de cada revisão, é estabelecida a ordem de verificação de sobreposição com as peças já posicionadas.

O Algoritmo 2 apresentado é denominado algoritmo de interação gravitacional otimizado IGO, porque corresponde a uma variação do algoritmo IG no qual são implementadas as hipóteses de otimização mencionadas acima. As hipóteses de movimentos com opção de otimização, uso de dicionários como memória e verificação de sobreposição em ordem da chance de ocorrência, são adicionadas à regra de posicionamento conforme mostrado neste algoritmo, nas linhas 25, 15 e 17 respectivamente. Da mesma forma que no Algoritmo 1, um critério de parada deve ser definido para delimitar a execução.

3.2.4 Regra de ordenação e estratégia de compactação

Na literatura tem sido estudado o efeito da ordenação da sequência de posicionamento das peças em técnicas de posicionamento consecutivo. Foi implementada a regra de ordenação de maior área primeiro, a qual obteve o melhor desempenho entre 8 regras testadas, no estudo realizado por Romanelli (2013), e foi chamado IGO_Ordem ao algoritmo resultante. Adicionalmente, tendo em conta a sequência do algoritmo IGO, que inicia o processo de compactação nos limites do comprimento e gradualmente é propagado para o interior do layout, foi implementada uma estratégia de alternância de posicionamento de peças e compactação, na etapa de construção da solução inicial. A estratégia foi inspirada em uma técnica similar presente no mesmo trabalho de Romanelli (2013).

A estratégia de alternância de posicionamento de peças e compactação consiste basicamente em dividir o conjunto total de peças em subconjuntos ou camadas, que são adicionadas iterativamente, resolvendo cada subproblema com o algoritmo IGO proposto, até posicionar e compactar todas as camadas e alcançar a solução inicial. Na Figura 17, observa-se o processo de compactação por camadas descrito, onde na coluna da esquerda apresenta-se o posicionamento inicial das peças para cada uma das camadas adicionadas ao layout, e na direita, observa-se o resultado da compactação do layout; parcial no caso das camadas 1 e 2, e da solução inicial, no caso da compactação da camada 3. Pos-

Algoritmo 2: IGO

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}
Saída: v, o

$Array_\mathcal{P}' \leftarrow$ Representação das peças;
$Array_D_{min} \leftarrow null$ ou $Matriz_D_{min}$;
$Array_IG \leftarrow null$;
$\Delta Y_{max} \leftarrow inf$;
enquanto *critério de parada não atingido* **faça**
    `/* Solução inicial e otimização */`
    enquanto *(o_{r1} em O_r) e (critério de parada não atingido)* **faça**
        enquanto *(P_p em $ind(\mathcal{P})$) e ($P_p > 1$)* **faça**
            $\{o_r, P_r, o_t\} \leftarrow Array_IG$;
            enquanto *(o_r em O_r) e ((não factível) ou ($\Delta Y \geq \Delta Y_{max}$))* **faça**
                enquanto *($P_r < P_p$) e ((não factível) ou ($\Delta Y \geq \Delta Y_{max}$))* **faça**
                    enquanto *(o_t em O_t) e ((não factível) ou ($\Delta Y \geq \Delta Y_{max}$))* **faça**
                        Verificação de restrições;
                        se *(factível) and ($\Delta Y < \Delta Y_{max}$)* **então**
                            $Array_IG \leftarrow \{o_r, P_r, o_t\}$;
                            `/* Atualizar em dicionário intervalos infactíveis */`
                            $Dicionário \leftarrow$ Intervalo infactível
                        senão
                            `/* Atualizar pontuação em lista para revisão de sobreposição */`
                            $Lista_revisão_sobreposição \leftarrow$ Atualizar pontuação
                        $o_t \leftarrow o_t + 1$;
                    $P_r \leftarrow P_r + 1$;
                $o_r \leftarrow o_r + 1$;
            se *((não factível) ou ($\Delta Y \geq \Delta Y_{max}$))* **então**
                $P_p \leftarrow P_p - 1$;
        se *(factível) e ($\Delta Y < \Delta Y_{max}$)* **então**
            $\Delta Y_{max}, v, o \leftarrow$ *Função(Array_IG)*;
            `/* Evitar movimentos sem opção de otimização */`
            enquanto *((($P_p > P_{LimSup}$) e ($P_p > P_{LimInf}$)) e ($P_p > 0$))* **faça**
                $P_p \leftarrow P_p - 1$;
        $o_r \leftarrow o_r + 1$;
    `/* Reordenar` $ind(\mathcal{P})$ `*/`
    Reordenar($ind(\mathcal{P})$);

teriormente, partindo da solução obtida, continua-se compactando o layout com o algoritmo IGO. O algoritmo que inclui a regra de ordenação e a estratégia de compactação por camadas foi chamado IGO_Ordem_Camadas, e sua sequência é apresentada no Algoritmo 3.

Para implementar a estratégia de compactação por camadas, deve-se estabelecer o critério de divisão do conjunto total de peças a serem posicionadas em cada iteração. No estudo de referência, estabelece-se um intervalo de peças, ou seja, define-se um número de peças a ser adicionado em cada iteração. Para este estudo, o critério foi estabelecido como uma porcentagem do conjunto total de peças, para manter uma proporção em instâncias de diferente porte, compactando sempre o mesmo número de camadas. Adicionalmente, da mesma forma que os algoritmos anteriores, um critério de parada deve ser definido para delimitar a execução, porém, esse algoritmo precisa de dois critérios de parada, um critério de parada local para delimitar o processo de compactação de cada camada, e um critério de parada global para finalizar a execução do algoritmo.

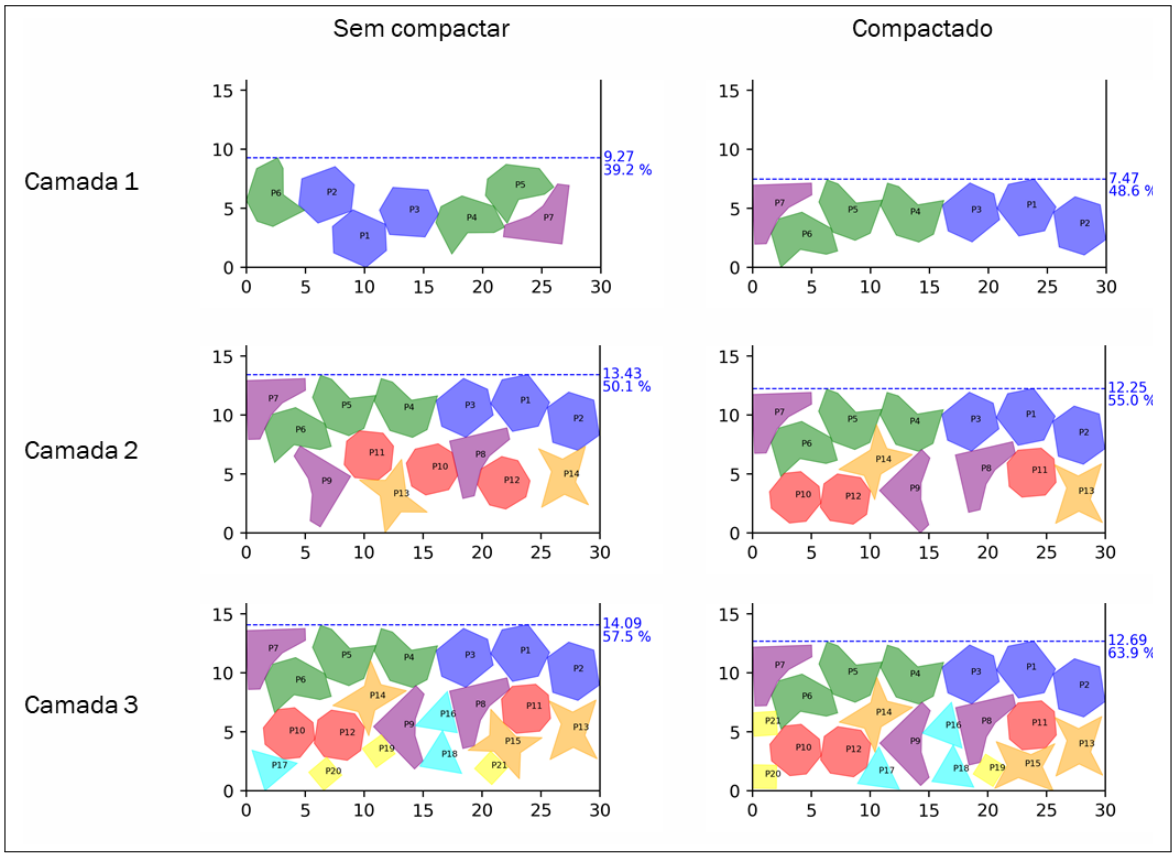


Figura 17: Exemplo de compactação por camadas. Fonte: Própria

Algoritmo 3: IGO_Ordem_Camadas

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}
Saída: v, o

$Array_\mathcal{P}' \leftarrow$ Representação das peças;
$Array_D_{min} \leftarrow null$ ou $Matriz_D_{min}$;
$Array_IG \leftarrow null$;
$\Delta Y_{max} \leftarrow inf$;
$ind(\mathcal{P}) \leftarrow$ maior_área_primeiro($ind(\mathcal{P})$);
$camadas \leftarrow$ dividir_em_subconjuntos($ind(\mathcal{P})$);
$sub_ind(\mathcal{P}) \leftarrow \{\}$;
enquanto *camada em camadas* **faça**
    $sub_ind(\mathcal{P}) \leftarrow sub_ind(\mathcal{P}) \cup camada$;
    enquanto *critério de parada local não atingido* **faça**
        IGO($\mathcal{P}'$, $sub_ind(\mathcal{P})$, ΔX_{max})
ΔY_{max}, v, $o \leftarrow$ *Função(Array_IG)*;
enquanto *critério de parada global não atingido* **faça**
    IGO($\mathcal{P}'$, $sub_ind(\mathcal{P})$, ΔX_{max})
ΔY_{max}, v, $o \leftarrow$ *Função(Array_IG)*;

Durante testes preliminares a estratégia de compactação por camadas apresentou uma melhoria significativa na solução inicial obtida e o algoritmo IGO na etapa posterior de otimização não apresentou um bom desempenho, aparentemente pela alta probabilidade combinatória ainda para a revisão do conjunto de soluções, e que parte de uma solução inicial com certo grau de compactação. Pelo

anterior, o algoritmo IGO_Ordem_Camadas foi destinado à etapa de construção da solução inicial, e foi simplificado para finalizar ao obter a primeira solução fatível, isto é, quando posiciona todas as camadas no layout como apresentado no Algoritmo 4.

3.2.5 Otimização com recozimento simulado

Visando avaliar de forma mais distribuída o espaço de soluções, foi implementada uma metaheurística de recozimento simulado, que permite aprimorar as soluções avaliando na vizinhança da solução atual e também permite escapar dos ótimos locais quando aceita soluções com maior custo, quando é simulado um incremento de temperatura.

O algoritmo de recozimento simulado proposto é inspirado no trabalho de Martins & Tsuzuki (2008). A Figura 18 permite explicar o posicionamento de uma peça nesta etapa de otimização. Em (a) é mostrada a solução atual a partir da qual é compactado do layout. Inicialmente é feito um retrocesso na sequência de posicionamento com a seleção aleatória de uma peça, desde a Peça 1 até a peça de maior índice que define os limites do layout, que corresponde à última peça com possibilidade de otimização de acordo com as hipóteses apresentadas na Seção 3.2.3; neste exemplo, da Peça 1 até a Peça 4. As peças com índice superior ou igual à peça selecionada são temporariamente eliminadas do layout, como observado na Figura 18(b), e o layout parcial apresenta um comprimento específico ($\Delta Y_{parcial}$), que é atribuído como valor inicial da variável de controle "temperatura" no algoritmo. Esta variável restringe o novo posicionamento da peça selecionada, aceitando valores de comprimento parcial menores ou iguais a este, e é utilizada ao longo do processo de recozimento para ajustar dinamicamente as condições de aceitação de novas configurações do layout.

As opções de novo posicionamento são dadas por operações permitidas de translação, rotação, incremento de temperatura e mudança de peça de referência, as quais se habilitam de forma sequencial como mostrado na Figura 18(d) e explicado no Algoritmo 5. Inicialmente, habilita-se a fase de translação da peça selecionada em relação à sua peça de referência P_r, variando em cada tentativa alfa (α) graus desde a translação do posicionamento atual, com α como ângulo aleatório entre o valor negativo e o valor positivo da máxima variação de translação permitida (veja Figura 18(d) item I)). Posteriormente, habilita-se a rotação da peça em relação ao seu próprio centroide, com uma variação

Algoritmo 4: IGO_Sol_Ini

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}
Saída: v, o

```
1  Array_P' ← Representação das peças;
2  Array_D_min ← null ou Matriz_D_min;
3  Array_IG ← null;
4  ΔY_max ← inf;
5  ind(P) ← maior_área_primeiro(ind(P));
6  camadas ← dividir_em_subconjuntos(ind(P));
7  sub_ind(P) ← {};
8  enquanto camada em camadas faça
9      sub_ind(P) ← sub_ind(P) ∪ camada;
10     enquanto critério de parada não atingido faça
11         IGO(P', sub_ind(P), ΔX_max)
12 ΔY_max, v, o ← Função(Array_IG);
```

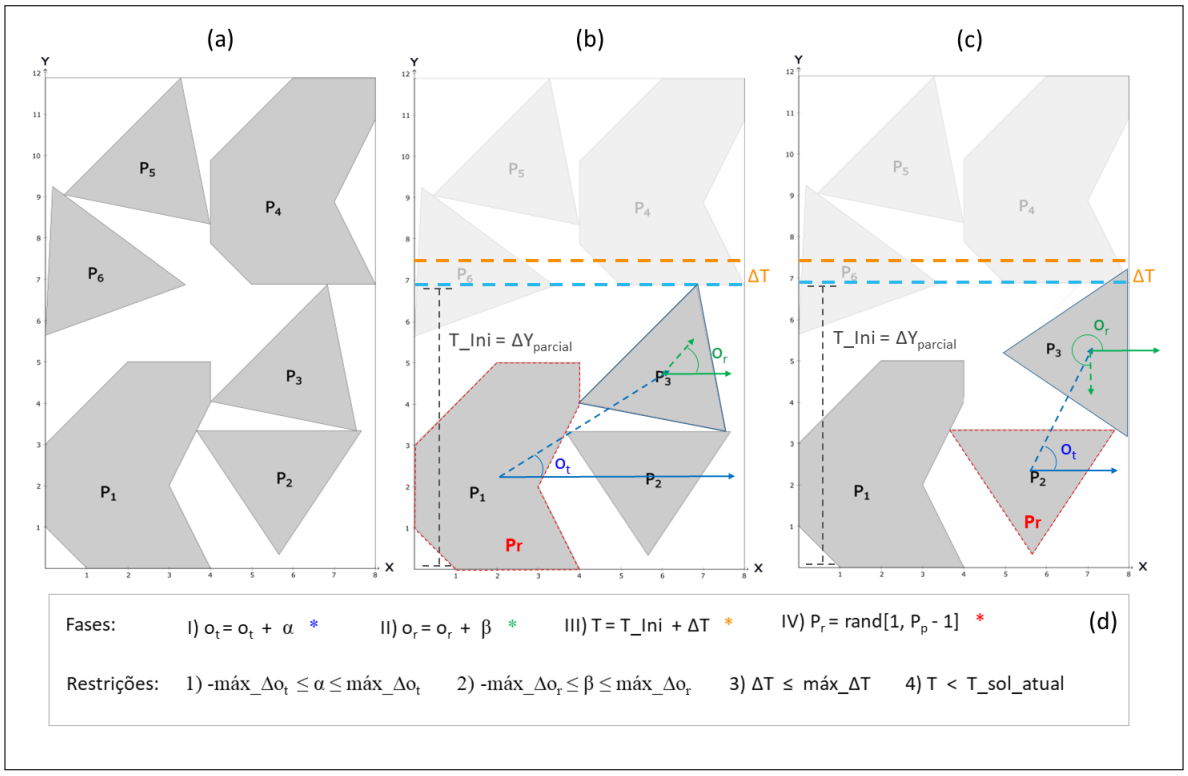


Figura 18: Exemplo de posicionamento de peça em etapa de otimização com recozimento simulado. (a) Solução atual. (b) Retrocesso aleatório na sequência de posicionamento e início das fases do recozimento. (c) Mudança de peça de referência. (d) Fases do recozimento e restrições. Fonte: Própria

permitida de beta (β) graus a partir da rotação do posicionamento atual, com β como ângulo aleatório entre o valor negativo e o valor positivo da máxima variação de rotação permitida (veja Figura 18(d) item II)). Como apresentado na Figura 18(d) item III), na terceira etapa da sequência de recozimento habilita-se o incremento gradual da temperatura (ΔT), que é um valor proporcional ao número de iterações subsequentes e vai desde 0 até o máximo incremento de temperatura permitido (máx_ΔT), com o qual a solução do novo layout parcial pode ter pior compactação que o layout parcial inicial ou de partida. A temperatura durante a etapa de recozimento nunca supera o valor da solução atual. Finalmente, habilita-se a seleção aleatória da peça de referência como é mostrado no exemplo da Figura 18(c), onde a nova peça de referência é a Peça 2. Os ângulos de rotação e translação a partir deste momento são escolhidos aleatoriamente, tomando valores inteiros entre 0 e 359. Continua-se aplicando o recozimento até posicionar a última peça.

Da mesma forma que os algoritmos anteriores, critérios de parada devem ser definidos para delimitar a execução. Esse algoritmo precisa de um critério de parada local para delimitar a execução das etapas do recozimento descrito (translação, rotação, incremento de temperatura e mudar de peça de referência) para tentar posicionar a peca escolhida no retrocesso aleatório, e de um critério de parada global. Neste trabalho, os critérios de parada locais foram estabelecidos por número de iterações e o critério de parada global, por limite de tempo.

Algoritmo 5: IGO_RS

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}
Saída: v, o

IGO_Sol_Ini($\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max});
enquanto *critério de parada global não atingido* **faça**
  $P_p \leftarrow$ Retrocesso aleatório com opção de otimização;
  enquanto *(P_p em $ind(\mathcal{P})$) e (critérios de parada global e local não atingidos)* **faça**
    $T = \Delta Y_{parcial}$
    enquanto *(Peça não posicionada) e (critérios de parada global e local não atingidos)* **faça**
      $\{o_r, P_r, o_t\} \leftarrow Array_IG$;
      se *(critério para translação é atingido)* **então**
        $o_t \leftarrow o_t + \alpha$;
      se *(critério para rotação é atingido)* **então**
        $o_r \leftarrow o_r + \beta$;
      se *(critério para incremento de temperatura é atingido)* **então**
        $T \leftarrow T + \Delta T$;
      se *(critério para mudar de peça de referência é atingido)* **então**
        $P_r \leftarrow$ Seleção aleatória;
        $o_t \leftarrow$ Seleção aleatória;
        $o_r \leftarrow$ Seleção aleatória;
      Verificação de restrições;
      se *(factível) and ($\Delta Y_parcial \leq T$)* **então**
        `/* A peça é posicionada */`
        $Array_IG \leftarrow \{o_r, P_r, o_t\}$;
      senão
        `/* Atualizar pontuação em lista para revisão de sobreposição */`
        $Lista_revisão_sobreposição \leftarrow$ Atualizar pontuação
    $P_p \leftarrow P_p + 1$;
  se *(factível) e ($\Delta Y < \Delta Y_{max}$)* **então**
    ΔY_{max}, v, $o \leftarrow$ *Função(Array_IG)*;

3.2.6 Otimização com escolha dinâmica de orientações permitidas

A técnica proposta consiste em habilitar um conjunto reduzido (subconjunto) de orientações permitidas para o posicionamento de cada uma das peças. Cada orientação tem associada uma pontuação que define o viés de seleção da orientação a ser substituída durante a alteração dinâmica do subconjunto de orientações. Na Figura 19, é apresentada a estratégia de implementação e um exemplo de configuração e inicialização.

Configuração (a): Consiste numa estrutura genérica de orientações permitidas para um conjunto

de peças $P_1, P_2, \ldots, P_n$. Para cada peça P_i, são configuradas múltiplas orientações Op_{i1}, Op_{i2}, Op_{i3}, $\ldots$, Op_{im}, onde m representa o número de orientações permitidas para cada peça. Os subconjuntos de orientações permitidas podem ser inicializados de forma aleatória, ou de forma ordenada e conveniente usando alguma regra ou critério na escolha.

Cada uma dessas orientações é inicialmente associada a uma pontuação de viés de seleção, representada por Po_{i1}, Po_{i2}, Po_{i3}, $\ldots$, Po_{im}. Esse viés de seleção define a chance de que uma orientação específica seja escolhida durante o processo de substituição dinâmica das orientações permitidas ao longo das iterações do algoritmo. Igualmente, a pontuação inicial para cada orientação pode ser estabelecida baseada em algum critério. Durante o processo de resolução do algoritmo, a pontuação dinâmica é atribuída às orientações com base na sua utilização nas soluções. Isso permite que o algoritmo explore eficientemente novas configurações potenciais ao adaptar e ajustar as orientações permitidas conforme o desenvolvimento na resolução.

Exemplo (b): Mostra um exemplo específico de inicialização dos subconjuntos de orientações permitidas. Nesse caso, as orientações foram definidas como os ângulos 0^o, 90^o, 180^o, 270^o, além de outros ângulos intermediários como 45^o, 135^o, 225^o e 315^o. As pontuações de viés de seleção atribuídas a essas orientações são mostradas na parte inferior da tabela, indicando que as orientações principais (0^o, 90^o, 180^o, 270^o) possuem uma pontuação de viés inicial maior (valor 10), enquanto os ângulos intermediários têm uma pontuação menor (valor 1). Isso significa que, inicialmente, o algoritmo tende a favorecer os ângulos retos ao tentar posicionar as peças. Ao longo das iterações essas pontuações de viés são atualizadas dinamicamente, dependendo do desempenho das orientações escolhidas.

O Algoritmo 6 inclui a implementação da técnica de escolha dinâmica de orientações permitidas,

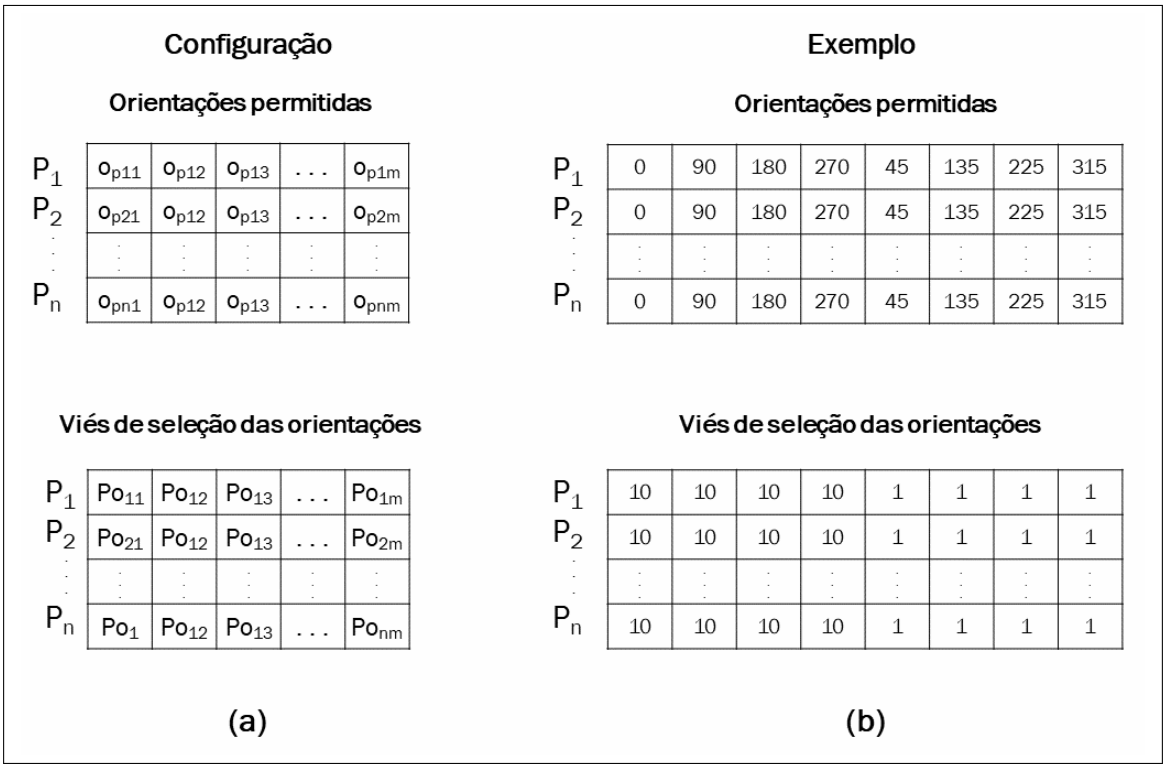


Figura 19: Estrutura dos arranjos para estratégia de escolha dinâmica de orientações permitidas. (a) configuração dos arranjos, (b) exemplo de parametrização inicial. Fonte: Própria

pelo qual é chamado de IGO_RS_OP, e difere pouco do algoritmo IGO_RS, acrescentando apenas a inicialização das orientações permitidas e do viés de seleção (nas linhas 1 e 2 respectivamente), a atualização de pontuações das orientações permitidas (na linha 8) e a escolha dinâmica de orientações (na linha 10).

Algoritmo 6: IGO_RS_OP

Entrada: $\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max}

Saída: v, o

1 $O_r \leftarrow$ Subconjuntos iniciais de orientações permitidas;
2 *Viés_de_seleção* $\leftarrow$ Pontuações iniciais das orientações permitidas;
3 IGO_Sol_Ini($\mathcal{P}'$, $ind(\mathcal{P})$, ΔX_{max});
4 **enquanto** *critério de parada global não atingido* **faça**
5 $P_p \leftarrow$ Retrocesso aleatório com opção de otimização;
`/* ... a mesma sequência do algoritmo IGO_RS */`
6 **se** *(factível) e* ($\Delta Y < \Delta Y_{max}$) **então**
7 ΔY_{max}, v, $o \leftarrow$ *Função(Array_IG)*;
8 *Viés_de_seleção* $\leftarrow$ Atualizar_pontuações(*Viés_de_seleção*, o);
9 **se** *(critério para alterar orientações permitidas é atingido)* **então**
10 $O_r \leftarrow$ Escolha_orientações(O_r, *Viés_de_seleção*);

CAPÍTULO 4

Testes computacionais

Este capítulo apresenta os testes computacionais que foram conduzidos para validar o método de resolução proposto. A melhoria dos algoritmos desenvolvidos ao longo do processo de formulação e aprimoramento do método foi conferido por meio da resolução de um conjunto de instâncias clássicas da literatura para o problema de corte e empacotamento irregular. As rotinas computacionais foram codificadas na linguagem de programação *Python* versão 3 e os testes foram realizados em um computador pessoal com processador Intel(R) Core(TM) i7-10510U 2.30GHz, sistema operacional Windows 11Pro de 64*bits* e memória RAM de 8GB.

4.1 Descrição das instâncias e desenho dos testes realizados

As instâncias resolvidas foram retiradas do repositório da ESICUP (CUTTING; PACKING, 2019) cujas características gerais são descritas na Tabela 4.1, considerando a nomenclatura utilizada por Bartmeyer et al. (2022), onde "TP" corresponde ao número total de tipos de peças, "d" é a demanda de cada tipo de peça e "m" é o número total de peças a serem posicionadas em cada instância. Além da nomenclatura da literatura, "ΔX_{max}" denota a dimensão fixa da faixa para cada caso. Quando "d" aparece na tabela representada por uma lista, cada posição da lista corresponde à quantidade do tipo

Tabela 4.1: Característica geral de cada instância

	Instância	TP	d	m	ΔX_{max}
original	Albano	8	$(2, 2, 4, 4, 4, 4, 2, 2)$	24	4.900
	Blaz	7	4 de cada tipo	28	15
	Jakobs	25	1 de cada tipo	25	40
	Jakobs2	25	1 de cada tipo	25	70
	Marques	8	$(4, 4, 2, 2, 4, 4, 2, 2)$	24	104
	Poly1a	15	1 de cada tipo	15	40
	Shirts	8	$(8, 8, 8, 15, 15, 15, 15, 15)$	99	40
modificada	RCO_n*	7	n de cada tipo	$7\times n$	15
	BLAZEWICZ_n*	7	n de cada tipo	$7\times n$	15

* Instâncias testadas usando n de 1 a 5.

de peça da instância correspondente aquela posição, caso contrário e por simplificação, a quantidade única para cada tipo de peça é apresentada. Em particular, nas instâncias RCO_n e BLAZEWICZ_n foi utilizado a letra "n", que foi variada entre 1 e 5 no testes, para indicar a quantidade de peças de cada tipo. Portanto, no total foram resolvidas 17 instâncias.

Observe ainda que as 7 primeiras instâncias na tabela são chamadas de originais, enquanto que as instâncias RCO_n e BLAZEWICZ_n são chamadas de modificadas, ou seja, elas são variações da instância Blaz comumente utilizadas na literatura para avaliar o comportamento das técnicas quando peças não convexas são incorporadas no problema. BLAZEWICZ_n incorpora os mesmos 7 tipos de peças da instância Blaz considerando a geometria original de cada peça, que pode variar entre convexas e não convexas. Em particular, observe na tabela que a instância BLAZEWICZ_n, para $n = 4$, coincide com a instância Blaz. No entanto, na apresentação dos resultados computacionais, elas foram tratadas como instâncias distintas por simplicidade na descrição dos resultados. Por outro lado, RCO_n emprega a envoltória convexa das mesmas peças utilizadas na BLAZEWICZ_n no lugar da geometria original de cada peça. Os resultados para RCO_n e BLAZEWICZ_n foram analisados e apresentados separadamente para, similarmente a literatura, avaliar o desempenho do método proposto quando peças não convexas aparecem no problema.

Os testes computacionais foram divididos em duas etapas principais. Na primeira etapa foram resolvidas todas as instâncias usando os algoritmos IG e IGO para uma avaliação inicial do desempenho da regra de posicionamento proposta, antes e depois de incluir a validação das hipóteses formuladas. A análise dos resultados nesta etapa é feita em dois momentos, a verificação do desempenho geral dos algoritmos na resolução das instâncias originais e do efeito de incluir peças não convexas no problema por meio da resolução das instâncias modificadas. Na segunda etapa, os testes computacionais foram realizados unicamente com as instâncias originais utilizando novas versões do algoritmo IGO que incluem as seguintes estratégias de otimização.

- **IGO_Ordem:** O critério do valor decrescente das áreas das peças é incorporado como regra de ordenação da sequência de posicionamento das peças.

- **IGO_Ordem_Camadas:** Na etapa de construção da solução inicial, a estratégia de compactação por camadas é incorporada no IGO_Ordem.

- **IGO_RS:** Na etapa de otimização da solução, a metaheurística de recozimento simulado proposta é incorporada no IGO_Ordem_Camadas. Note que IGO_RS é uma simplificação de IGO_Ordem_Camadas_RS.

- **IGO_RS_OP:** A estratégia de escolha dinâmica das orientações permitidas é incorporada no IGO_RS.

O tempo de execução de cada algoritmo durante a resolução de cada instância foi limitado em 1.800 segundos, valor apropriado quando comparado com o tempo dos melhores resultados relatados na literatura. Neste trabalho, todo o tempo gasto com pré-processamento e todos os processos realizados para resolver o problema estão incluídos no tempo de execução, incluindo a construção em cada iteração

do arranjo $Array_D_{min}$ que armazena as distâncias de colisão entre peças e o pré-processamento (inferior a 5 segundos) para construir o arranjo $Array_\mathcal{P}'$ que representa as peças.

4.2 Resultados para os algoritmos IG e IGO

4.2.1 Desempenho em relação as instâncias originais

A Tabela 4.2 apresenta os resultados obtidos pelos algoritmos IG e IGO na resolução das instâncias originais do repositório ESICUP, sem incluir as instâncias RCO$_n$ e BLAZEWICZn. A tabela exibe a melhor solução encontrada por cada algoritmo, a eficiência alcançada % Efi. (calculada como a área total das peças posicionadas dividida pela área do objeto, multiplicada por 100), e o tempo necessário para alcançar essa solução (Tempo [s]). A coluna tSolIG indica o tempo que o algoritmo IGO levou para encontrar a melhor solução que foi obtida pelo algoritmo IG dentro do tempo limite. Observa-se que a eficiência das soluções obtidas pelo algoritmo IGO superou em média a do IG em 7,2 pontos percentuais, com destaque para a instância Jakobs, onde a melhoria foi de 13,6 pontos percentuais.

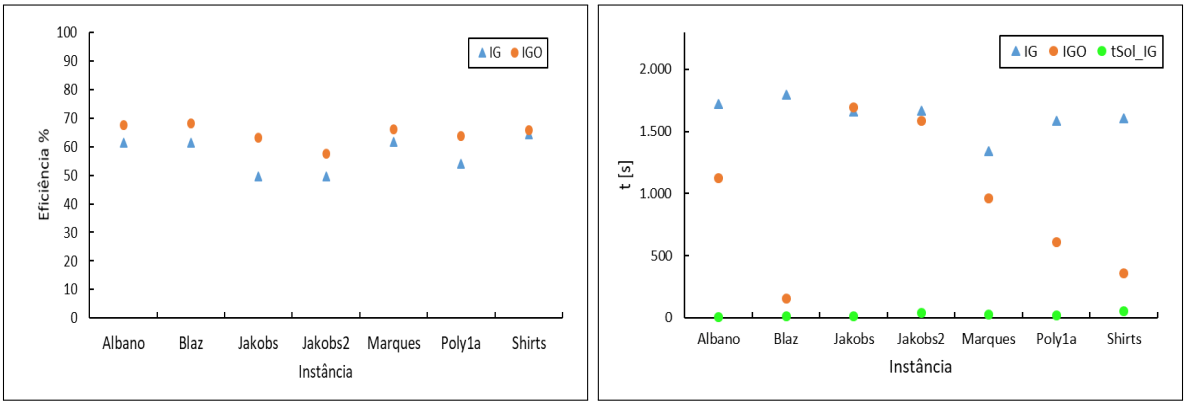
Tabela 4.2: Resultados dos algoritmos IG e IGO para as instâncias originais

	IG			IGO			
Instância	Melhor Sol.	% Efi.	Tempo [s]	Melhor Sol.	% Efi.	Tempo [s]	tSol$_{IG}$ [s]
Albano	14.184,12	61,37	1.723,14	12.902,19	67,47	1.125,85	2,91
Blaz	35,11	61,52	1.794,87	31,66	68,22	155,42	8,63
Jakobs	19,82	49,45	1.657,40	15,53	63,10	1.694,43	12,96
Jakobs2	38,94	49,56	1.664,41	33,51	57,59	1.587,58	39,38
Marques	112,21	61,65	1.342,43	104,67	66,09	963,57	25,20
Poly1a	19,02	53,89	1.582,07	16,07	63,78	608,01	20,43
Shirts	83,90	64,36	1.608,65	81,91	65,93	353,56	55,02

A Figura 20 ilustra a porcentagem de eficiência alcançada e o tempo computacional gasto nas melhores soluções obtidas por cada um dos algoritmos, com o objetivo de observar a otimização que pode ser obtida aplicando as hipóteses propostas. Os resultados mostram que para todas as instâncias o algoritmo otimizado IGO encontrou uma solução melhor que o algoritmo básico IG, que é o resultado esperado, tendo em conta que neste momento o algoritmo IGO não omite nenhuma solução, mas sim acelera o processamento delas e omite aquelas que não são viáveis, através de hipóteses.

A Figura 4.20(b) mostra a otimização do tempo computacional alcançada pelo algoritmo IGO em relação ao algoritmo IG, onde a série IG representa o tempo gasto para obter a melhor solução com seu respectivo algoritmo, e a série tSol$_{IG}$, o tempo gasto para obter essa mesma solução com o algoritmo IGO. A otimização média do tempo computacional foi de 98%. Considerando que, após 25% do tempo decorrido, instâncias como Blaz e Shirts não mostram melhoria na solução, é provável que a compactação posterior na mesma sequência tenha pouco potencial de avanço ou que o custo computacional que precisa, supere os limites estabelecidos.

A instância Shirts foi a instância que apresentou menor otimização com a implementação do algoritmo IGO, apenas 1,5%, porém esta é a instância com maior número de peças para posicionar (99 peças), o que indica que dada a alta probabilidade combinatória ainda não é possível revisar o conjunto



(a) Eficiência

(b) Tempo computacional.

Figura 20: Eficiência e tempo computacional das melhores soluções obtidas pelos algoritmos IG e IGO para as instâncias originais. Fonte: Própria

de soluções de forma eficiente, com a primeira otimização do algoritmo.

4.2.2 Desempenho em relação as instâncias modificadas

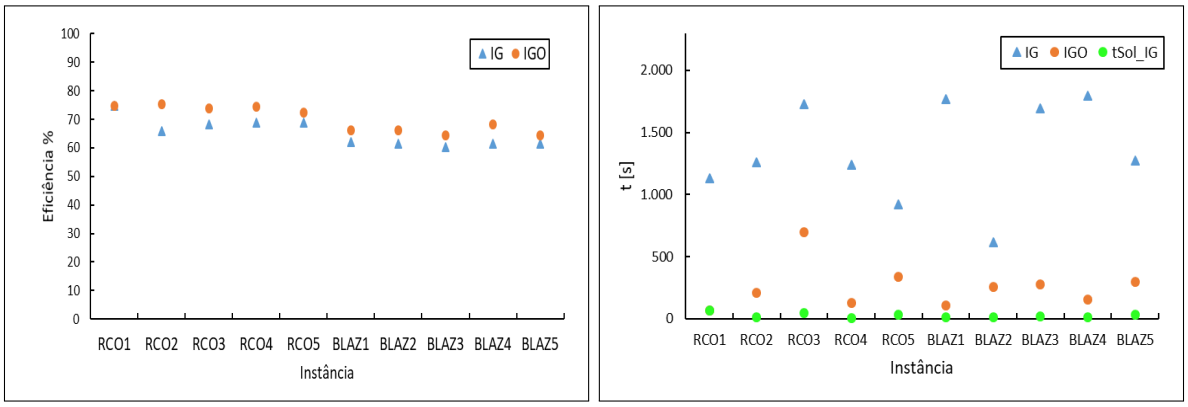
A Tabela 4.3 apresenta os resultados obtidos com os algoritmos IG e IGO na resolução das variações das instâncias RCO_n e BLAZEWICZ_n, ou seja, quando $1 \leq n \leq 5$. As colunas desta tabela compartilha a mesma estrutura da Tabela 4.2.

Tabela 4.3: Resultados dos algoritmos IG e IGO para as instâncias RCO_n e BLAZEWICZ_n

	IG			IGO			
Instância	Melhor Sol.	% Efi.	Tempo [s]	Melhor Sol.	% Efi.	Tempo [s]	tSol_{IG} [s]
RCO_1	8,43	74,73	1.128,00	8,43	74,73	68,08	68,08
RCO_2	19,11	65,93	1.257,63	16,74	75,27	211,44	10,94
RCO_3	27,75	68,11	1.727,12	25,60	73,83	699,48	42,43
RCO_4	36,70	68,66	1.241,25	33,89	74,36	126,84	6,44
RCO_5	45,74	68,87	920,48	43,47	72,46	335,75	29,41
BLAZEWICZ_1	8,70	62,07	1.770,01	8,17	66,10	103,37	11,02
BLAZEWICZ_2	17,59	61,40	612,95	16,29	66,30	257,15	11,54
BLAZEWICZ_3	26,89	60,25	1.691,42	25,18	64,34	277,41	17,15
BLAZEWICZ_4	35,11	61,52	1.794.87	31,66	68,22	155.42	8.63
BLAZEWICZ_5	43,88	61,53	1274,21	41,84	64,54	297,39	34,91

A Figura 21 mostra a eficiência e o tempo computacional gasto das melhores soluções dos algoritmos para as instâncias RCO_n e BLAZEWICZ_n. Além de avaliar a otimização pelas hipóteses propostas, analisa-se o comportamento dos algoritmos com peças convexas e não convexas, aproveitando que RCO_n representa a envoltória convexa das instâncias BLAZEWICZ_n. A Figura 4.21(a) não mostra diferença representativa na eficiência alcançada na implementação dos algoritmos nos dois tipos de instâncias, mesmo com a variação no número de peças. Por outro lado, na Figura 4.21(b) não existe relação entre o tempo computacional necessário para obter a melhor solução e o tipo de instância. O que foi dito acima é uma indicação de que para a técnica proposta o processamento de peças não convexas não é uma limitação ou desvantagem. Nestas instâncias a otimização média do tempo computacional,

para obter com o algoritmo IGO a melhor solucao obtida pelo algoritmo IG, também foi de 98%.



(a) Eficiência (b) Tempo computacional.

Figura 21: Eficiência e tempo computacional das melhores soluções obtidas pelos algoritmos IG e IGO para as instâncias em RCO_n e BLAZEWICZ_n. Fonte: Própria

4.3 Resultados para as modificações do algoritmo IGO nas instâncias originais

Esta seção adiciona a análise dos resultados para as modificações do algoritmo IGO descritas na Seção 4.1, ou seja, os algoritmos IGO_Ordem, IGO_Ordem_Camadas, IGO_RS e IGO_RS_OP. Particularmente, essas modificações foram testadas apenas nas instâncias originais (veja Tabela 4.1). Na Figura 22 é possível observar o comportamento da eficiência alcançada no tempo pelos algoritmos IG, IGO, IGO_Ordem e IGO_Ordem_Camadas, testados nas diferentes instâncias. Para todas as instâncias, exceptuando a instância Blaz, a implementação das hipóteses, a regra de ordenação e a estratégia de compactação por camadas, foram sequencialmente melhorando a solução alcançada. Em Blaz a regra de ordenação não apresentou uma melhoria na solução alcançada e posteriormente a implementação do posicionamento por camadas também não apresentou melhoria. Como o posicionamento por camadas implementado inclui a regra de ordenação, foi feito um teste adicional com a instância Blaz para avaliar o desempenho do posicionamento por camadas independente da regra de ordenação, e para observar se o baixo desempenho nos dois casos foi por causa da regra de ordenação. Como se observa na Figura 26 do Anexo 5, a implementação independente da estratégia de compactação consegue melhorar a solução alcançada pelo algoritmo IGO e se evidencia então que a regra de ordenação implementada tem efeito negativo para a instância Blaz particularmente.

Para olhar melhor o comportamento dos algoritmos na etapa da construção da solução inicial e no início da otimização, são apresentadas na Figura 23 os gráficos dos primeiros 60 segundos de execução dos algoritmos, onde se observa na maioria dos casos melhoria nas soluções iniciais alcançadas pelas estratégias implementadas e em todos os casos, o algoritmo IGO_Ordem_Camadas alcançou uma melhor solução inicial do que os outros, no entanto para todas as instâncias o algoritmo não melhorou a solução depois dos 900 segundos, como observado na Figura 22, com o qual inferiu-se que o algoritmo IGO_Ordem_Camadas tem potencial para obter soluções iniciais de qualidade, mas que precisa de técnicas adicionais na etapa de otimização. Devido a isso, foi simplificado o algoritmo para

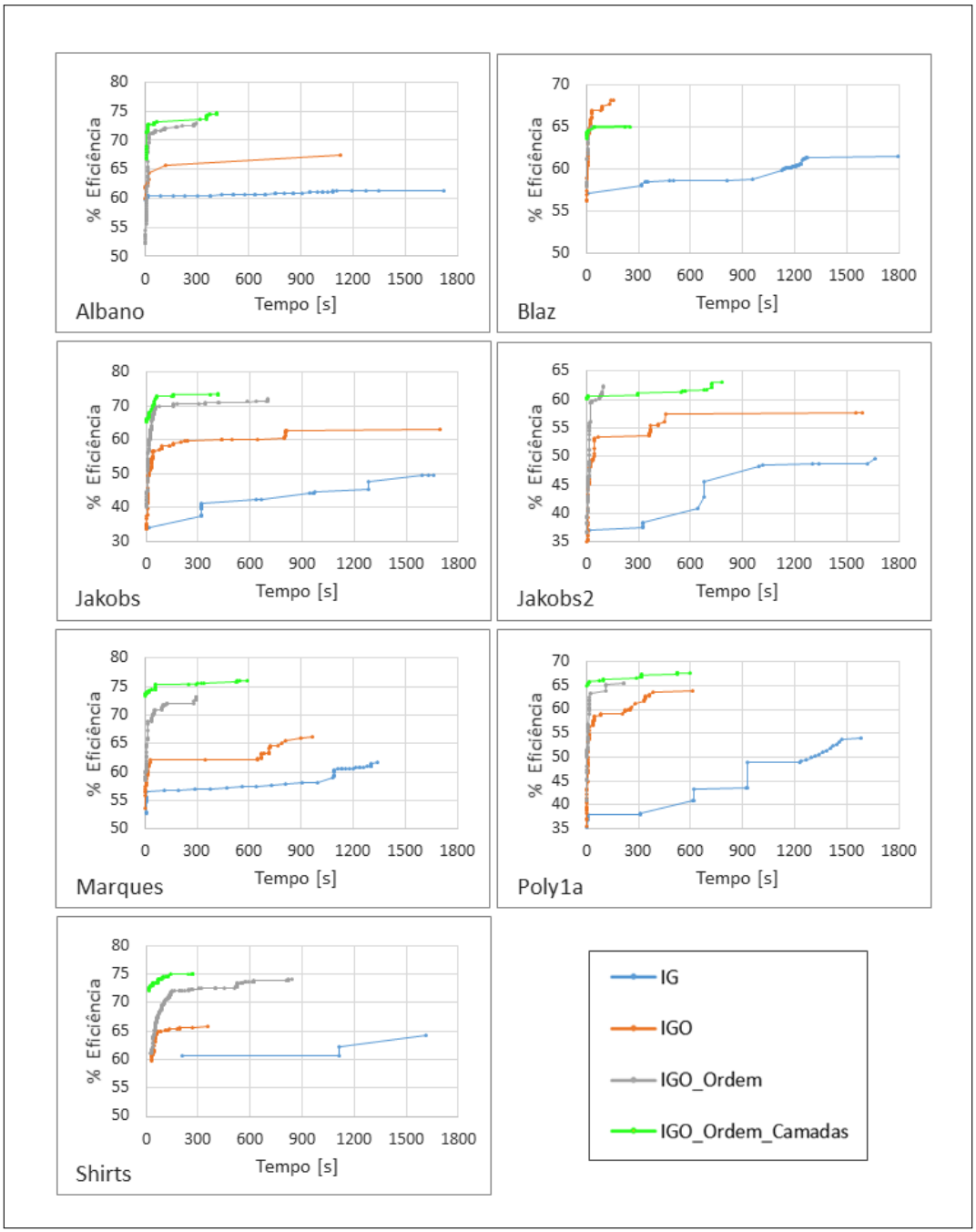


Figura 22: Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Ordem_Camadas. Fonte: Própria

alcançar somente a solução inicial e foi chamado IGO_Sol_Ini conforme apresentado na Seção 3.2.4 do Capítulo 3.

A implementação da meta-heurística de recozimento simulado na etapa de otimização, melhorou em todas as instâncias as soluções alcançadas com o algoritmo IGO_Ordem_Camadas. Além disso, aproveitou melhor o tempo de execução, indicando uma exploração mais distribuída e contínua do conjunto de soluções, como observado na Figura 24, onde é apresentada a eficiência obtida no tempo com o algoritmo IGO_Ordem_Camadas e durante uma das execuções do algoritmo IGO_RS, porque este último tem escolha aleatória de variáveis durante o processo e as soluções exploradas são diferentes em cada execução.

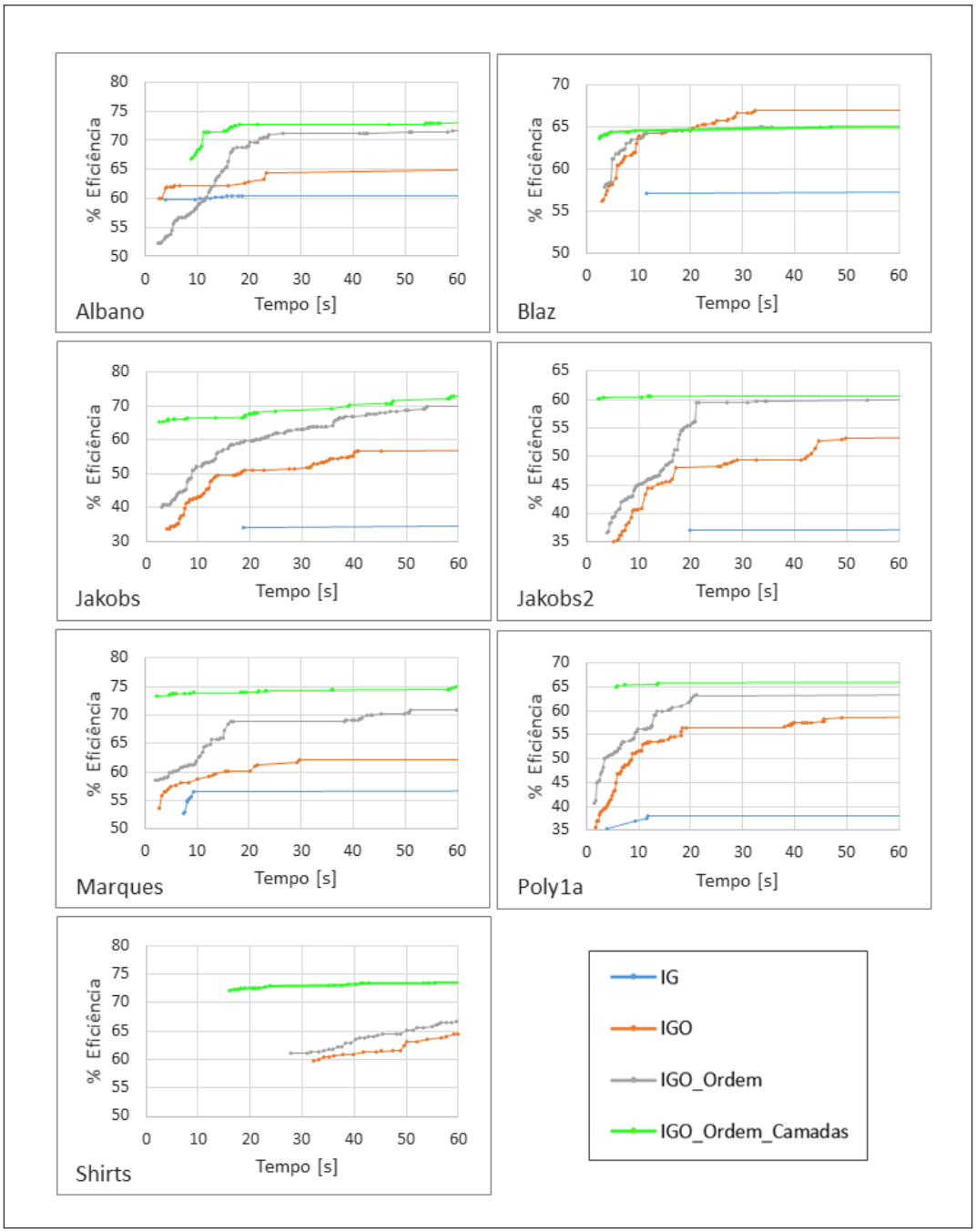


Figura 23: Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Ordem_Camadas nos primeiros 60 segundos. Fonte: Própria

Na Figura 25 é apresentado um comparativo das eficiências no tempo alcançadas pelos algoritmos IGO_RS_OP e IGO_RS, durante um dos processamentos das instâncias testadas. Adicionalmente, apresenta a melhor solução encontrada na literatura, de técnicas de resolução que permitem orientações livres (linha descontinua verde) e de técnicas que permitem somente um conjunto discreto de orientações (linha descontinua cinza). Em todas as instâncias o algoritmo IGO_RS_OP obteve uma melhor solução inicial, e também um melhor desempenho no início da etapa de otimização. Nas únicas instâncias que não superou o resultado final do algoritmo IGO_RS foi em Blaz e Jakobs2, e nas instancias Jakobs, Marques e Poly1a, superou as técnicas encontradas que resolvem o problema com orientações livres.

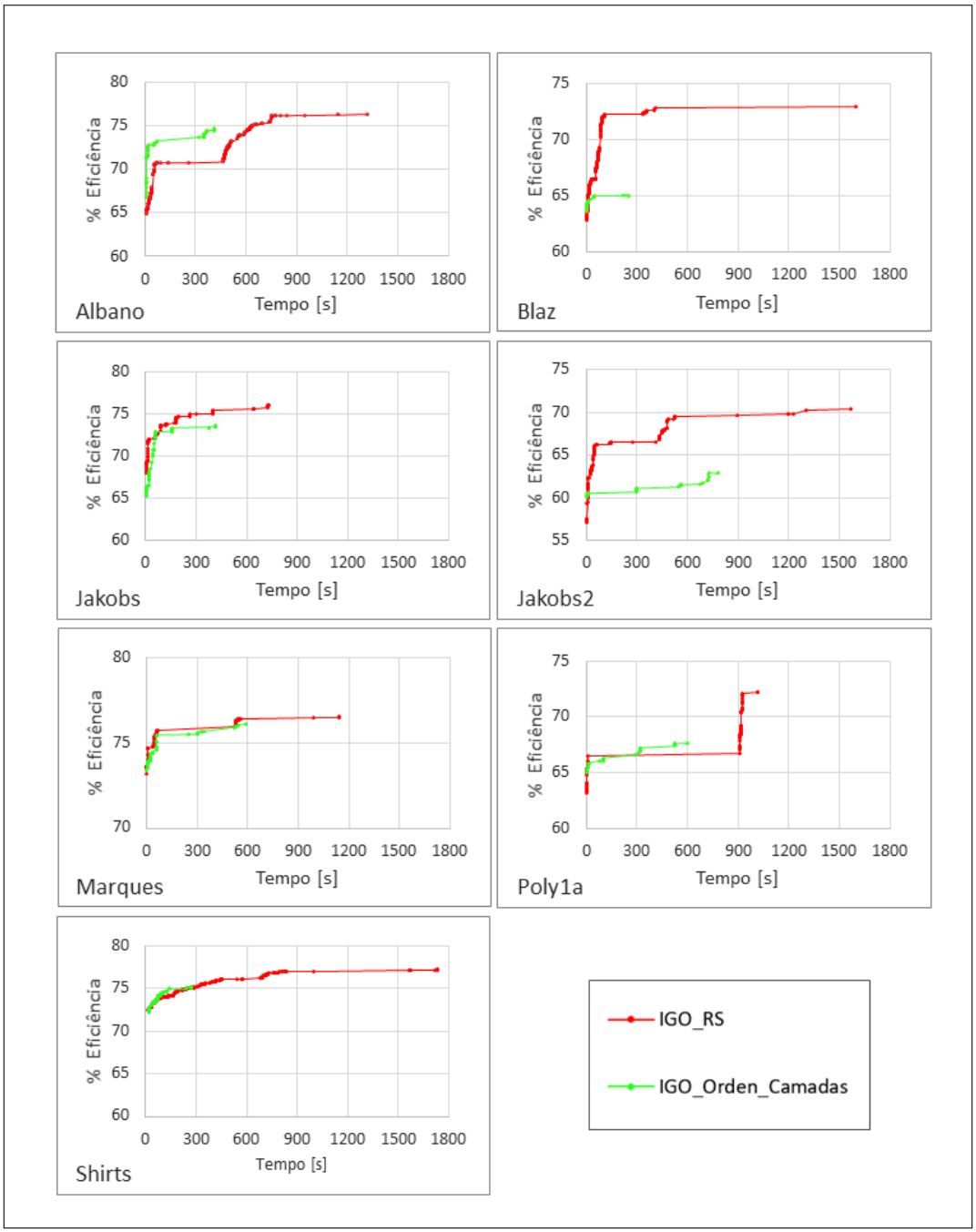


Figura 24: Eficiência no tempo dos algoritmos IGO_RS e IGO_Ordem_Camadas. Fonte: Própria

A Tabela 4.4 apresenta os melhores resultados obtidos com o algoritmo IGO_RS_OP na resolução das diferentes instâncias testadas, e os melhores resultados encontrados na literatura para essas instâncias. Na maioria das instâncias o melhor resultado foi obtido por um algoritmo da literatura que permite orientações discretas, unicamente na instância Blaz obteve melhor desempenho uma técnica da literatura para orientações livres. Tanto nos gráficos, quanto na tabela de comparação, se observa que na maioria dos casos, as orientações discretas não limitaram aos algoritmos para alcançar melhores soluções que os algoritmos que permitem orientações livres, evidenciando que as formulações discretas foram mais eficientes na revisão de seu conjunto de soluções, e que a estratégia de reduzir o conjunto de orientações permitidas pode apresentar um caminho para melhorar o desempenho dos algoritmos

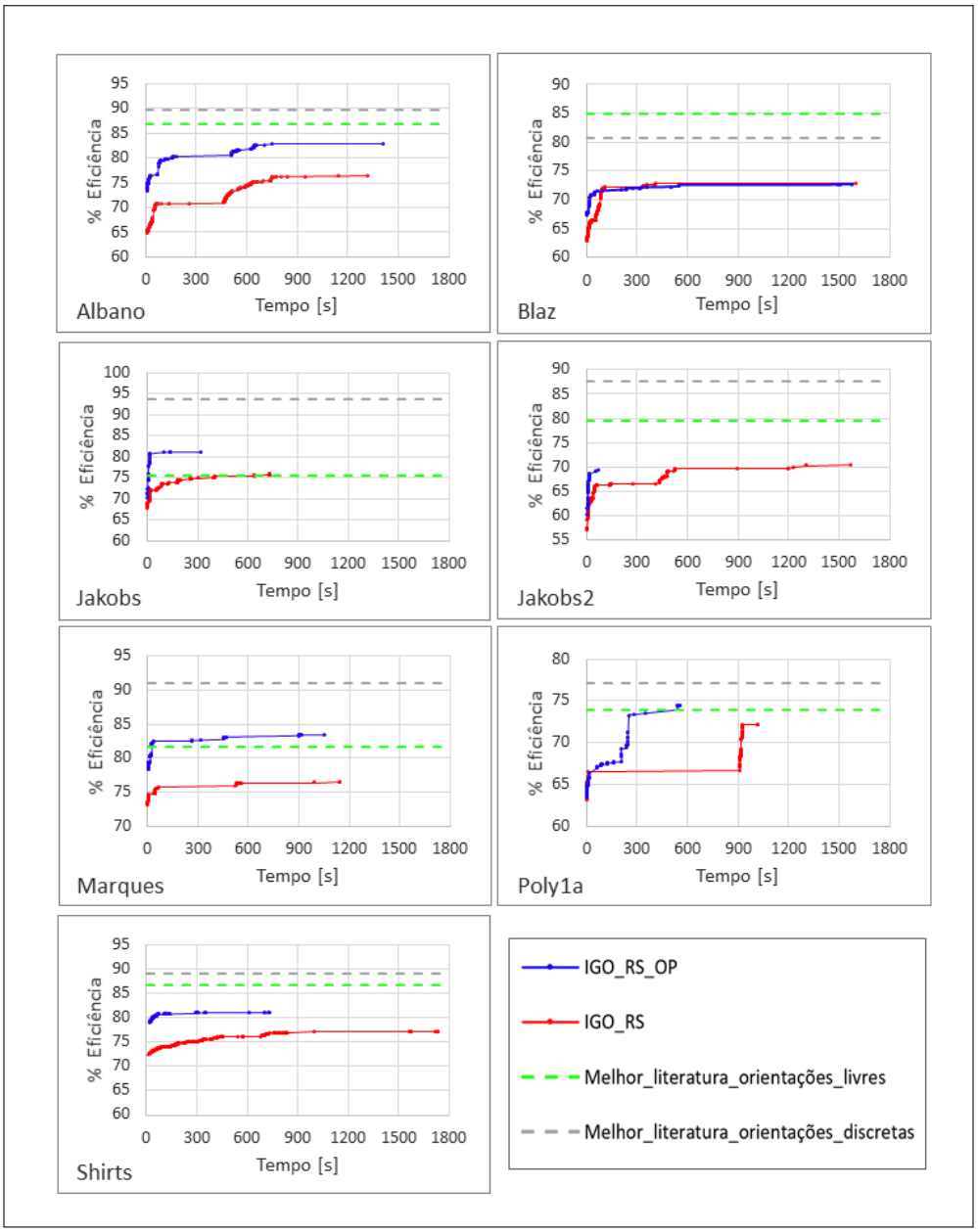


Figura 25: Eficiência no tempo dos algoritmos IGO_RS_OP e IGO_RS. Fonte: Própria

com orientações livres.

O Anexo 5 contém figuras com resultados gráficos entregues pelos algoritmos IGO_RS e IGO_RS_OP ao final de sua execução, para as diferentes instâncias testadas. Em cada figura, a sub-figura (a) corresponde ao resultado obtido pelo algoritmo IGO_RS, e a sub-figura (b) corresponde ao resultado obtido pelo algoritmo IGO_RS_OP. À direita de cada sub-figura pode-se observar, de cima para baixo, o comprimento, a porcentagem de eficiência e o tempo gasto, da solução obtida.

Tabela 4.4: Comparação dos melhores resultados entre IGO_RS_OP e aqueles da literatura (retirados de Burke et al. (2010), Elkeran (2013), Peralta et al. (2018), Sato et al. (2019), Sato et al. (2023))

	Melhor IGO_RS_OP		Melhor lit. livres		Melhor lit. discretas		
Instância	% Efi.	Tempo [s]	% Efi.	Tempo [s]	% Efi.	Tempo [s]	Orientações (Graus)
Albano	82,91	1.412,09	86,77	124,39	**89,58**	1.200,00	0 e 180
Blaz	72,64	1.572,17	**85,01**	25,42	80.60	300,00	0 e 180
Jakobs	81,26	318,80	75,44	23,98	**93,78**	1800,00	0, 90, 180 e 270
Jakobs2	69,55	68,72	79,59	53,67	**87,73**	600,00	0, 90, 180 e 270
Marques	83,46	1.053,94	81,72	124,85	**91,02**	1.200,00	0, 90, 180 e 270
Poly1a	74,52	553,75	73,90	27,59	**77,07**	3.600,00	0
Shirts	81,16	729,52	86,83	1.820,28	**88,96**	1.200,00	0 e 180

4.4 Síntese dos resultados computacionais

Os resultados dos testes computacionais para os algoritmos IG e IGO indicam que a inclusão da validação das hipóteses formuladas (movimentos com opção de otimização, Otimização com uso de dicionários como memória e verificação de sobreposição em ordem da chance de ocorrência) reduz o conjunto de configurações (layout) a verificar e, com isso, melhora o desempenho computacional do algoritmo. Adicionalmente, os resultados dos testes computacionais para as modificações do algoritmo IGO mostram que as diferentes estratégias de otimização induzem a uma melhoria no desempenho dos algoritmos na maioria das instâncias, porém, em algumas instâncias, a regra de ordenação empregada não melhora o desempenho.

O algoritmo IGO_RS_OP performou melhor significativamente na grande maioria das instâncias, sendo que apenas na instância Blaz, ele apresentou uma solução ligeiramente inferior à solução alcançada pelo algoritmo IGO_RS em relação a eficiência do espaço ocupado no objeto, que foi inferior 0,33%. Além disso, em 3 instâncias, o algoritmo IGO_RS_OP também obteve melhores resultados quando comparado com os resultados encontrados na literatura para orientações livres das peças como pode ser conferido na Tabela 4.4 e Figura 25.

CAPÍTULO 5

Conclusões

Nesta dissertação foram estudadas diferentes ferramentas computacionais utilizadas na resolução de problemas de corte e empacotamento em faixas, a partir das quais foi possível propor um novo método de resolução. O foco principal das técnicas estudadas é a simplificação do processo de verificação de sobreposição por meio de estratégias de representação da geometria das peças e do objeto. Além disso, a revisão da literatura revela que o uso de modelos matemáticos apropriados é importante na resolução do problema, inclusive para derivar representações geométricas.

A principal contribuição deste trabalho envolveu o desenvolvimento e validação de regras de alocação de peças, que foram evoluindo gradualmente ao satisfazer um conjunto de formulações de hipóteses e por meio da incorporação de técnicas da literatura ou de estratégias inspiradas nelas. Essa evolução derivou o algoritmo IGO_RS_OP como sendo o representante do método final de resolução proposto. Adicionalmente, uma importante contribuição deste trabalho foi o desenvolvimento de uma técnica que usa a distância mínima entre duas peças (distância de colisão) no posicionamento e na verificação de sobreposição. Nela, as distâncias calculadas ao longo da rotina são aproveitados entre iterações.

Vale a pena observar que inicialmente foi formulada uma regra de posicionamento que lembra a atratividade física de corpos e progressivamente ela foi modificada para alcançar uma exploração mais eficiente do conjunto de soluções factíveis, realizado em um processo dinâmico de melhoria, o que a torna diferente das metodologias tradicionais de resolução. Os resultados obtidos mostram que a abordagem proposta neste trabalho é bastante promissora. Ela aponta que com o uso de formulações de hipóteses adequadas e com uma implementação abrangente de técnicas de otimização é possível desenvolver regras de posicionamento eficientes.

A síntese dos resultados computacionais revelam o bom desempenho do método proposto e também sugere potencial para avanços na metodologia por meio de ajustes variados nas configurações dos parâmetros nas diferentes rotinas. Então, a exploração de métodos combinando as diferentes modificações propostas para o algoritmo IGO e de ajustes dos parâmetros poderiam configurar trabalhos futuros interessantes. Adicionalmente, existem possibilidades de incorporação e adaptação de outras técnicas da literatura, como o aprendizado de máquina e processamento computacional paralelo.

Além das contribuições diretas, este trabalho abre espaço para a exploração de novos modelos

matemáticos que utilizem as técnicas desenvolvidas, como a verificação de sobreposição baseada na distância de colisão. Essa abordagem pode ser integrada a formulações matemáticas mais robustas, proporcionando um novo campo de pesquisa para a resolução de problemas de empacotamento irregular com maior eficiência.

Bibliografia

Arenales, M. N., Armentano, V. A., Morabito Neto, R. & Yanasse, H. H. (2007), 'Pesquisa operacional'.

Babu, A. R. & Babu, N. R. (2001), 'A generic approach for nesting of 2-d parts in 2-d sheets using genetic and heuristic algorithms', *Computer-Aided Design* **33**(12), 879–891.

Baldacci, R., Boschetti, M. A., Ganovelli, M. & Maniezzo, V. (2014), 'Algorithms for nesting with defects', *Discrete Applied Mathematics* **163**, 17–33.

Bartmeyer, P. M., Oliveira, L. T., Leão, A. A. S. & Toledo, F. M. B. (2022), 'An expert system to react to defective areas in nesting problems', *Expert Systems with Applications* **209**, 118207.

Bennell, J. A. & Oliveira, J. F. (2008), 'The geometry of nesting problems: A tutorial', *European journal of operational research* **184**(2), 397–415.

Bennell, J. A. & Oliveira, J. F. (2009), 'A tutorial in irregular shape packing problems', *Journal of the Operational Research Society* **60**(sup1), S93–S105.

Burke, E. K., Hellier, R. S., Kendall, G. & Whitwell, G. (2007), 'Complete and robust no-fit polygon generation for the irregular stock cutting problem', *European Journal of Operational Research* **179**(1), 27–49.

Burke, E. K., Hellier, R. S., Kendall, G. & Whitwell, G. (2010), 'Irregular packing using the line and arc no-fit polygon', *Operations Research* **58**(4-part-1), 948–970.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M., Oliveira, J. F. & Carravilla, M. A. (2016), 'Robust mixed-integer linear programming models for the irregular strip packing problem', *European Journal of Operational Research* **253**(3), 570–583.

Diyaley, S. & Chakraborty, S. (2022), 'Metaheuristics-based nesting of parts in sheet metal cutting operation', *Operational Research in Engineering Sciences: Theory and Applications* **5**(2), 1–16.

Domovic, D., Rolich, T., Grundler, D. & Bogovic, S. (2014), Algorithms for 2d nesting problem based on the no-fit polygon, *in* '2014 37th international convention on information and communication technology, electronics and microelectronics (MIPRO)', IEEE, pp. 1094–1099.

Dyckhoff, H. & Finke, U. (1992), *Cutting and packing in production and distribution: A typology and bibliography*, Springer Science & Business Media.

El Naqa, I. & Murphy, M. J. (2015), *What is machine learning?*, Springer.

Elkeran, A. (2013), 'A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering', *European Journal of Operational Research* **231**(3), 757–769.

Gahm, C., Uzunoglu, A., Wahl, S., Ganschinietz, C. & Tuma, A. (2022), 'Applying machine learning for the anticipation of complex nesting solutions in hierarchical production planning', *European Journal of Operational Research* **296**(3), 819–836.

Gomes, A. M. (2013), 'Irregular packing problems: Industrial applications and new directions using computational geometry', *IFAC Proceedings Volumes* **46**(7), 378–383.

Gomes, A. M. & Oliveira, J. F. (2002), 'A 2-exchange heuristic for nesting problems', *European Journal of Operational Research* **141**(2), 359–370.

Holland, J. H. (1992), *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT press.

Kimms, A. & Király, H. (2023), 'An extended model formulation for the two-dimensional irregular strip packing problem considering general industry-relevant aspects', *European Journal of Operational Research* **306**(3), 1202–1218.

Kirkpatrick, S., Gelatt Jr, C. D. & Vecchi, M. P. (1983), 'Optimization by simulated annealing', *science* **220**(4598), 671–680.

Konopasek, M. (1981), 'Mathematical treatments of some apparel marking and cutting problems', *US Department of Commerce Report* **99**(26), 90857–10.

Leão, A. A. S. & Toledo, F. M. B. d. (2021), 'Enhanced solution for the irregular strip packing problem: valid inequalities and branching priorities', *Anais* .

Leão, A. A., Toledo, F. M., Oliveira, J. F. & Carravilla, M. A. (2016), 'A semi-continuous mip model for the irregular strip packing problem', *International Journal of Production Research* **54**(3), 712–721.

Leão, A. A., Toledo, F. M., Oliveira, J. F., Carravilla, M. A. & Alvarez-Valdés, R. (2020), 'Irregular packing problems: A review of mathematical models', *European Journal of Operational Research* **282**(3), 803–822.

Liao, X., Ma, J., Ou, C., Long, F. & Liu, X. (2016), 'Visual nesting system for irregular cutting-stock problem based on rubber band packing algorithm', *Advances in Mechanical Engineering* **8**(6), 1687814016652080.

Liu, C., Si, Z., Hua, J. & Jia, N. (2023), 'Optimizing two-dimensional irregular packing: a hybrid approach of genetic algorithm and linear programming', *Applied Sciences* **13**(22), 12474.

Martins, T. d. C. & Tsuzuki, M. S. (2008), 'Rotational placement of irregular polygons over containers with fixed dimensions using simulated annealing and no-fit polygons', *Journal of the Brazilian Society of Mechanical Sciences and Engineering* **30**, 205–212.

Mundim, L. R., Andretta, M. & de Queiroz, T. A. (2017), 'A biased random key genetic algorithm for open dimension nesting problems using no-fit raster', *Expert Systems with Applications* **81**, 358–371.

Nunes, W. H. (2021), 'Algoritmos heurísticos para o problema de nesting com rotações livres [dissertação de mestrado]', *Universidade Federal de Lavras* .

Oliveira, J. F. C. & Ferreira, J. A. S. (1993), 'Algorithms for nesting problems', *Applied simulated annealing* pp. 255–273.

Oliveira, J. F., Gomes, A. M. & Ferreira, J. S. (2000), 'Topos–a new constructive algorithm for nesting problems: Topos–ein neues konstruktionsverfahren für das „nesting-problem"', *OR-Spektrum* **22**, 263–284.

Peralta, J., Andretta, M. & Oliveira, J. F. (2018), 'Solving irregular strip packing problems with free rotations using separation lines', *Pesquisa Operacional* **38**, 195–214.

Qi, D., Yang, W., Ding, L., Wu, Y., Tian, C., Yuan, L., Wang, Y. & Huang, Z. (2023), 'An intelligent approach to the unit nesting problem of coil material', *Applied Sciences* **13**(16), 9067.

Rocha, P., Rodrigues, R., Gomes, A. M., Toledo, F. M. & Andretta, M. (2014), 'Circle covering representation for nesting problems with continuous rotations', *IFAC Proceedings Volumes* **47**(3), 5235–5240.

Rodrigues, M. O. & Toledo, F. M. (2017), 'A clique covering mip model for the irregular strip packing problem', *Computers & Operations Research* **87**, 221–234.

Romanelli, A. (2013), Proposal for a heuristic approach for generation of two-dimensional irregular cutting layout, Dissertação (mestrado), Universidade Federal de Viçosa, Viçosa. Mestrado em Metodologias e técnicas da Computação; Sistemas de Computação.

Rutenbar, R. A. (1989), 'Simulated annealing algorithms: An overview', *IEEE Circuits and Devices magazine* **5**(1), 19–26.

Sato, A. K., Martins, T. C., Gomes, A. M. & Tsuzuki, M. S. G. (2019), 'Raster penetration map applied to the irregular packing problem', *European Journal of Operational Research* **279**(2), 657–671.

Sato, A. K., Mundim, L. R., Martins, T. C. & Tsuzuki, M. S. G. (2023), 'A separation and compaction algorithm for the two-open dimension nesting problem using penetration-fit raster and obstruction map', *Expert Systems with Applications* **220**, 119716.

Segenreich, S. A. & Braga, L. M. P. F. (1986), 'Optimal nesting of general plane figures: a monte carlo heuristical approach', *Computers & Graphics* **10**(3), 229–237.

Stoyan, Y., Scheithauer, G., Gil, N. & Romanova, T. (2004), 'ϕ-functions for complex 2d-objects', *Quarterly Journal of the Belgian, French and Italian Operations Research Societies* **2**, 69–84.

Stoyan, Y., Terno, J., Scheithauer, G., Gil, N. & Romanova, T. (2002), 'Phi-functions for primary 2d-objects', *Stud. Inform. Univ.* **2**(1), 1–32.

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement learning: An introduction*, MIT press.

Toledo, F. M., Carravilla, M. A., Ribeiro, C., Oliveira, J. F. & Gomes, A. M. (2013), 'The dotted-board model: a new mip model for nesting irregular shapes', *International Journal of Production Economics* **145**(2), 478–487.

Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K. & Ibaraki, T. (2009), 'Solving the irregular strip packing problem via guided local search for overlap minimization', *International Transactions in Operational Research* **16**(6), 661–683.

Wang, A., Hanselman, C. L. & Gounaris, C. E. (2018), 'A customized branch-and-bound approach for irregular shape nesting', *Journal of Global Optimization* **71**, 935–955.

Wäscher, G., Haußner, H. & Schumann, H. (2007), 'An improved typology of cutting and packing problems', *European journal of operational research* **183**(3), 1109–1130.

Yau, H.-T. & Hsu, C.-W. (2022), 'Nesting of 3d irregular shaped objects applied to powder-based additive manufacturing', *The International Journal of Advanced Manufacturing Technology* **118**(5), 1843–1858.

Anexo 1: Modelos matemáticos

Este capítulo apresenta uma descrição do caso geral do problema de empacotamento irregular, denotado como *Nesting Problem* (NP). Além disso, alguns exemplos de modelagem matemática para uma versão do NP é apresentado. Esta dissertação considera o NP como um problema de empacotamento bidimensional em que um conjunto de peças deve ser posicionado em um contêiner (objeto) considerando um layout factível. O formato geométrico das peças e do objeto pode ser irregular, regular ou ambos. Por exemplo, o formato pode ser representado por polígonos simples, sendo que em vários casos quando o polígono não é simples, ele poderia ser decomposto em polígonos simples. O objetivo é obter uma solução factível que minimize o espaço não ocupado do objeto na solução. As restrições geométricas consideram que i) uma peça não deve se sobrepor a outras peças e ii) as peças devem ficar completamente dentro do objeto. A notação apresentada por Umetani et al. (2009) é usada para descrever a seguinte modelagem matemática para o NP. Seja P um polígono de m arestas representado pela lista de seus vértices $(x_1, y_1), \ldots, (x_m, y_m)$. Considere um conjunto de polígonos simples $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ representando a geometria de todas as peças a serem posicionadas. O objeto é representado por C, cuja geometria é definida em cada tipo de problema de empacotamento – o caso mais comum considera um objeto retangular. Orientações são permitidas para cada peça, podendo envolver orientações padronizadas como 0°, 90°, 180° e 270°, um conjunto discreto com uma maior variedade de orientações e, em alguns casos, a orientação é livre.

Uma peça com uma orientação específica é denotada por $P(o)$, onde o é o ângulo de rotação de P. O interior de P é representado por $int(P)$. $P \oplus v$ corresponde à aplicação de uma translação (soma de Minkowski) pelo vetor v em P. O conjunto de orientações permitidas é denotado por O (comumente aparece contendo múltiplos de 90°). Uma solução para o NP é apresentada por um conjunto de vetores de translação $v_1, \ldots, v_n$ e um conjunto de orientações permitidas $o_1, \ldots, o_n$ e deve satisfazer dois tipos distintos de restrições.

1. Se as peças i e j são posicionadas no mesmo objeto, então elas não podem apresentar sobreposição: $int(P_i(o_i) \oplus v_i) \cap int(P_j(o_j) \oplus v_j) = \emptyset$.

2. Se uma peça i é posicionada em um objeto, então ela deve estar completamente contida no objeto: $(P_i(o_i) \oplus v_i) \subseteq C$.

A seguir aparece uma modelagem para o caso particular do problema de empacotamento irregular em faixas. Neste caso, o objeto é representado por um retângulo semi-infinito C, o qual tem a largura fixa e o comprimento variável. O objetivo é obter uma solução viável que maximize a densidade da configuração, ou seja, que minimize o comprimento do layout. Seja L o comprimento do menor objeto que contém todas as peças. O seguinte modelo descreve o problema de empacotamento irregular em faixas.

$$\begin{aligned}
\text{Minimizar} \quad & L && (1)\\
\text{sujeito a} \quad & \text{int}(P_i(o_i) \oplus v_i) \cap \text{int}(P_j(o_j) \oplus v_j) = \emptyset, \quad 1 \leq i < j \leq n, && (2)\\
& (P_i(o_i) \oplus v_i) \subseteq \mathcal{C}, \quad 1 \leq i \leq n, && (3)\\
& o_i \in \mathcal{O}, \quad 1 \leq i \leq n, && (4)\\
& v_i \in \mathbb{R}^2, \quad 1 \leq i \leq n, && (5)\\
& L \in \mathbb{R}_+. && (6)
\end{aligned}$$

Note que a Figura 1 ilustra uma solução para o Modelo (1)–(6). Toledo et al. (2013) desenvolveram outra modelagem interessante, para a qual variáveis de decisão binárias são associadas a cada ponto discreto do objeto e a cada tipo de peça. O objeto (ou tabuleiro) é retangular, tem largura fixa e um limite superior para o comprimento deve ser calculado conforme a quantidade de peças da instância. Os índices $p, u \in \mathcal{P}$ indicam os tipos de peças, os índices $c \in C$ e $r \in R$ indicam as colunas e as linhas da discretização do objeto, respectivamente. O índice $d \in D$ representa cada célula da discretização do objeto, que também pode ser representado por um par de valores $(c, r) \in C \times R$, no qual $c = \lfloor d/|R| \rfloor$ e $r = d - \lfloor d/|R| \rfloor \times |R| - 1$ e, reciprocamente, $d = c \times |R| + r + 1$. A resolução da grade na dimensão x e y são representadas por g_x e g_y, respectivamente. Cada tipo de peça p é representada por um polígono definido por um conjunto de vértices cujas coordenadas são relativas ao ponto de referência: i) $(0, 0)$ ponto de referência de p; ii) $(x_p^1, y_p^1), (x_p^2, y_p^2), \ldots, (x_p^{v_p}, y_p^{v_p})$ é a lista de vértices da peça p, dada no sentido horário; iii) $x_p^m, x_p^M, y_p^m, y_p^M$ referem-se aos limites do envelope da peça p; iv) q_p é o número de peças do tipo p que devem ser posicionadas. Utiliza-se o conceito do polígono de ajuste interno ($\mathcal{IFP}_p$) entre o tipo de peça p e o objeto, para garantir que se verificam as restrições geométricas no seguinte modelo matemático.

$$\begin{aligned}
\text{Minimizar} \quad & L && (7)\\
\text{sujeito a} \quad & (C \times g_x + x_p^M) \times \delta_p^d \leq L, \quad d \in \mathcal{IFP}_p,\ p \in \mathcal{P}, && (8)\\
& \sum_{d \in \mathcal{IFP}_p} \delta_p^d = q_p, \quad p \in \mathcal{P}, && (9)\\
& \delta_u^e + \delta_p^d \leq 1, \quad e \in \mathcal{NFP}_{p,u}^d,\ p, u \in \mathcal{P},\ d \in \mathcal{IFP}_p, && (10)\\
& \delta_p^d \in \{0, 1\}, \quad d \in \mathcal{IFP}_p,\ p \in \mathcal{P}, && (11)\\
& L \geq 0, && (12)
\end{aligned}$$

onde o objetivo é minimizar o comprimento do objeto L que contém todas as peças. O comprimento L é definido conforme as Restrições (8). Para garantir que todas as peças estejam posicionadas utiliza-se as Restrições (9). A não sobreposição é garantida pelas Restrições (10). As Restrições (11) definem

as variáveis de decisão apenas para as posições onde as peças estão dentro do objeto. Finalmente, o domínio da variável L é dado pela Restrição (12).

Peralta et al. (2018) apresentaram uma modelagem envolvendo uma formulação não linear para o problema de empacotamento de peças irregulares, podendo ser convexas ou não convexas e tendo rotação livre. O objeto onde são colocadas as peças tem largura fixa e comprimento ilimitado. A representação das n peças a serem posicionadas é realizada de tal forma que uma peça é particionada em peças convexas $P_i = [P_{i,1}, P_{i,2}, \ldots, P_{i,p_i}]$, com $i \in \{1, \ldots, n\}$ e sendo p_i o número de peças convexas pertencentes à partição da peça P_i, de modo que uma partição da peça P_i e denotada por $P_{i,j}$ e as coordenadas de um vértice pertencente a essa partição em sua posição original são dadas por $(x^l_{i,j}, y^l_{i,j})$, com $j \in \{1, \ldots, p_i\}$ e $l \in \{1, \ldots, u_{i,j}\}$, sendo $u_{i,j}$ o número de vértices da partição convexa j da peça P_i. A não sobreposição é garantida aplicando trigonometria direita, especificamente utilizando linhas de separação dadas pela equação $y = c_{i,r}x + d_{i,r}$. Cada par de peças $P_{i,j}$ e $P_{r,s}$ tem obrigatoriamente uma linha de separação pertencente à partição das peças P_i e P_r, com $i \neq r$. Seja $(L, q_1, q_2, \ldots, q_n, \overline{q}_1, \overline{q}_2, \ldots, \overline{q}_Q)$ o vetor de todas as variáveis do modelo, sendo L o comprimento do objeto, q_i as variáveis referentes à peça P_i, $q_i = (\overline{x}^1_i, \overline{y}^1_i, \theta_i)$ com $i \in \{1, \ldots, n\}$ onde $(\overline{x}^1_i, \overline{y}^1_i)$ e θ_i são, respectivamente, as variáveis do ponto de referencia da peça P_i e de seu ângulo de rotação, e sendo $\overline{q}_\ell$ as variáveis referentes à reta que separa a peça $P_{i,j}$ da peça $P_{r,s}$, $\overline{q}_\ell = (\overline{x}_{i,j,r,s}, \overline{y}_{i,j,r,s}, \alpha_{i,j,r,s})$, com $\ell \in \{1, \ldots, Q\}$ e sendo $Q = \sum_{i=1}^{n-1} p_i(\sum_{k=i+1}^{n} p_k)$ o número de linhas de separação. $(\overline{x}_{i,j,r,s}, \overline{y}_{i,j,r,s})$ é o ponto de referencia da reta separadora e $\alpha_{i,j,r,s}$ é o seu ângulo de rotação.

Sejam $N = \{1, \ldots, n\}$, $J_i = \{1, \ldots, p_i\}$, $U_{i,j} = \{1, \ldots, u_{i,j}\}$, e a largura fixa do objeto e L o seu comprimento, o modelo matemático é o seguinte.

$$\text{Minimizar} \quad L \tag{13}$$

$$\text{sujeito a} \quad 0 \leq \overline{y}^l_{i,j} \leq e, \quad i \in N,\ j \in J_i,\ l \in U_{i,j} \tag{14}$$

$$0 \leq \overline{x}^l_{i,j} \leq L, \quad i \in N,\ j \in J_i,\ l \in U_{i,j}, \tag{15}$$

$$\overline{y}^l_{i,j} - c_{i,j,r,s}\overline{x}^l_{i,j} - d_{i,j,r,s} \leq 0, \quad i, r \in N\ i \neq r,\ j \in J_i,\ s \in J_r,\ l \in U_{i,j}, \tag{16}$$

$$\overline{y}^l_{r,s} - c_{i,j,r,s}\overline{x}^l_{r,s} - d_{i,j,r,s} \geq 0, \quad i, r \in N\ i \neq r,\ j \in J_i,\ s \in J_r,\ l \in U_{i,j}, \tag{17}$$

$$L \geq 0, \tag{18}$$

$$\text{onde:} \quad c_{i,j,r,s} = \frac{(x_{k+1} - x_k)\sin\alpha_{i,j,r,s} + (y_{k+1} - y_k)\cos\alpha_{i,j,r,s}}{(x_{k+1} - x_k)\cos\alpha_{i,j,r,s} + (y_k - y_{k+1})\sin\alpha_{i,j,r,s}} \tag{19}$$

$$d_{i,j,r,s} = \bar{y}_k - c_{i,j,r,s}\bar{x}_k, \tag{20}$$

$$\bar{x}^l_{i,j} = x^l_{i,j}\cos\theta_i - y^l_{i,j}\sin\theta_i + \bar{x}^1_i, \tag{21}$$

$$\bar{y}^l_{i,j} = x^l_{i,j}\sin\theta_i + y^l_{i,j}\cos\theta_i + \bar{y}^1_i, \tag{22}$$

(x_k, y_k) e (x_{k+1}, y_{k+1}) são os dois vértices de uma das peças pelos quais passa a reta que separa as duas peças.

O objetivo é minimizar L que corresponde ao máximo valor de $\overline{x}^l_{i,j}$, $L = max\{\overline{x}^l_{i,j}\}$. As Restrições (14) e (15) garantem que todas as peças $P_{i,j}$ estejam inteiramente dentro do objeto. As Restri-

ções (16) e (17) garantem a não sobreposição entre as peças. O domínio da variável L é dado pela Restrição (18).

Anexo 2: Teste adicional para a instância Blaz

A Figura 26 apresenta os resultados de um teste adicional realizado com a instância Blaz, onde não é implementada a regra de ordenação da sequência de posicionamento das peças durante a implementação da estratégia de compactação por camadas, para avaliar o desempenho do posicionamento por camadas independente da regra de ordenação.

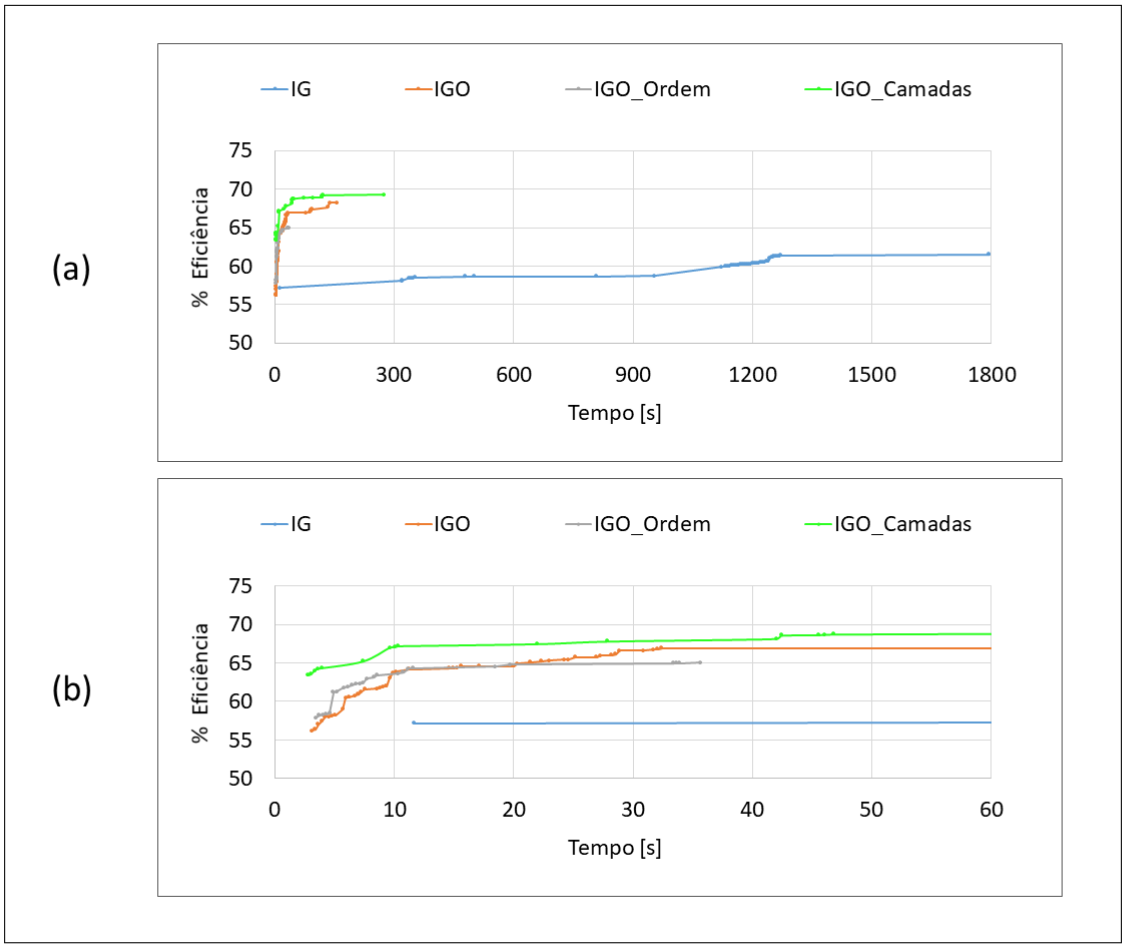
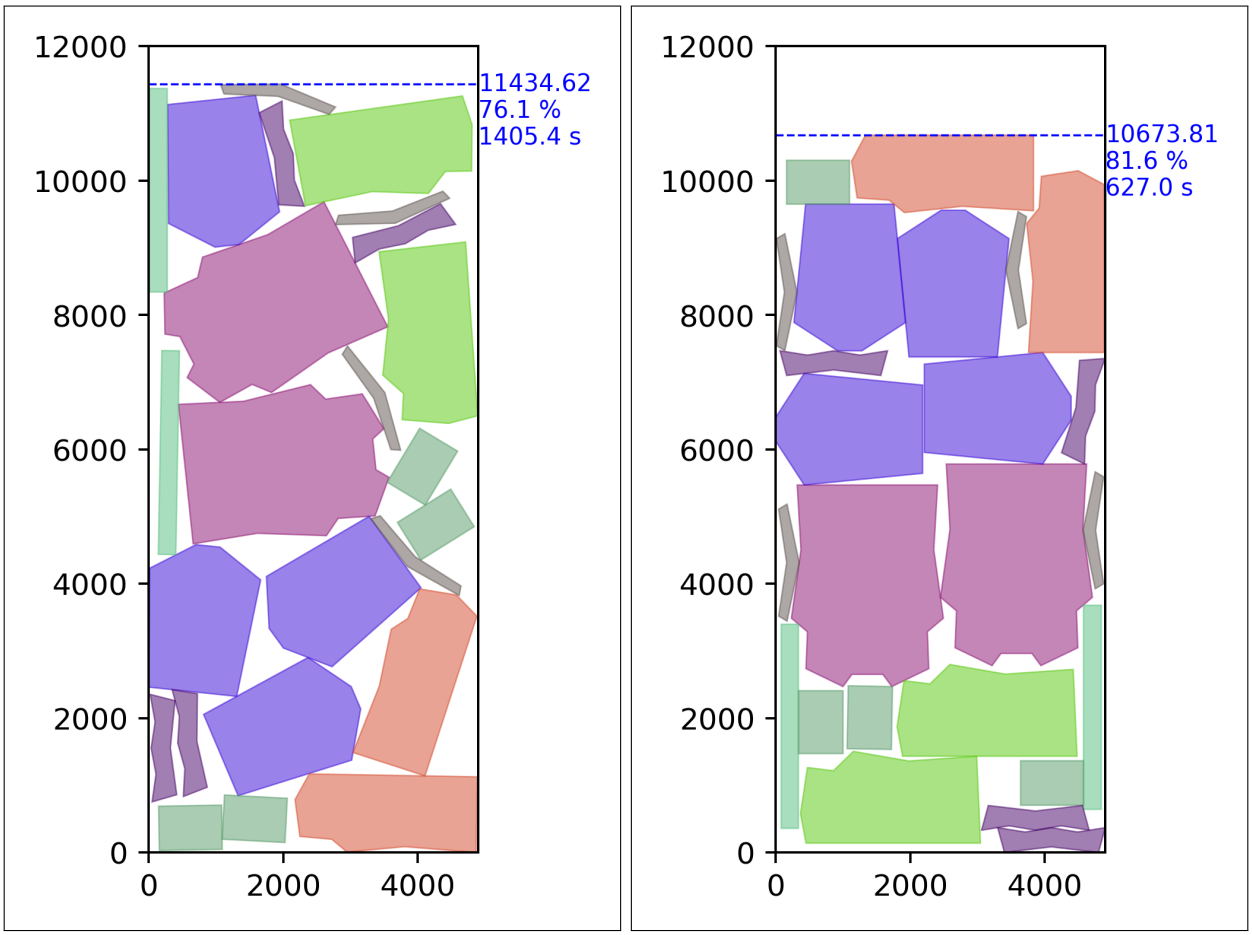


Figura 26: Eficiência no tempo dos algoritmos IG, IGO, IGO_Ordem e IGO_Camadas: a) no tempo limite de 1800 segundos; e b) nos primeiros 60 segundos. Fonte: Própria

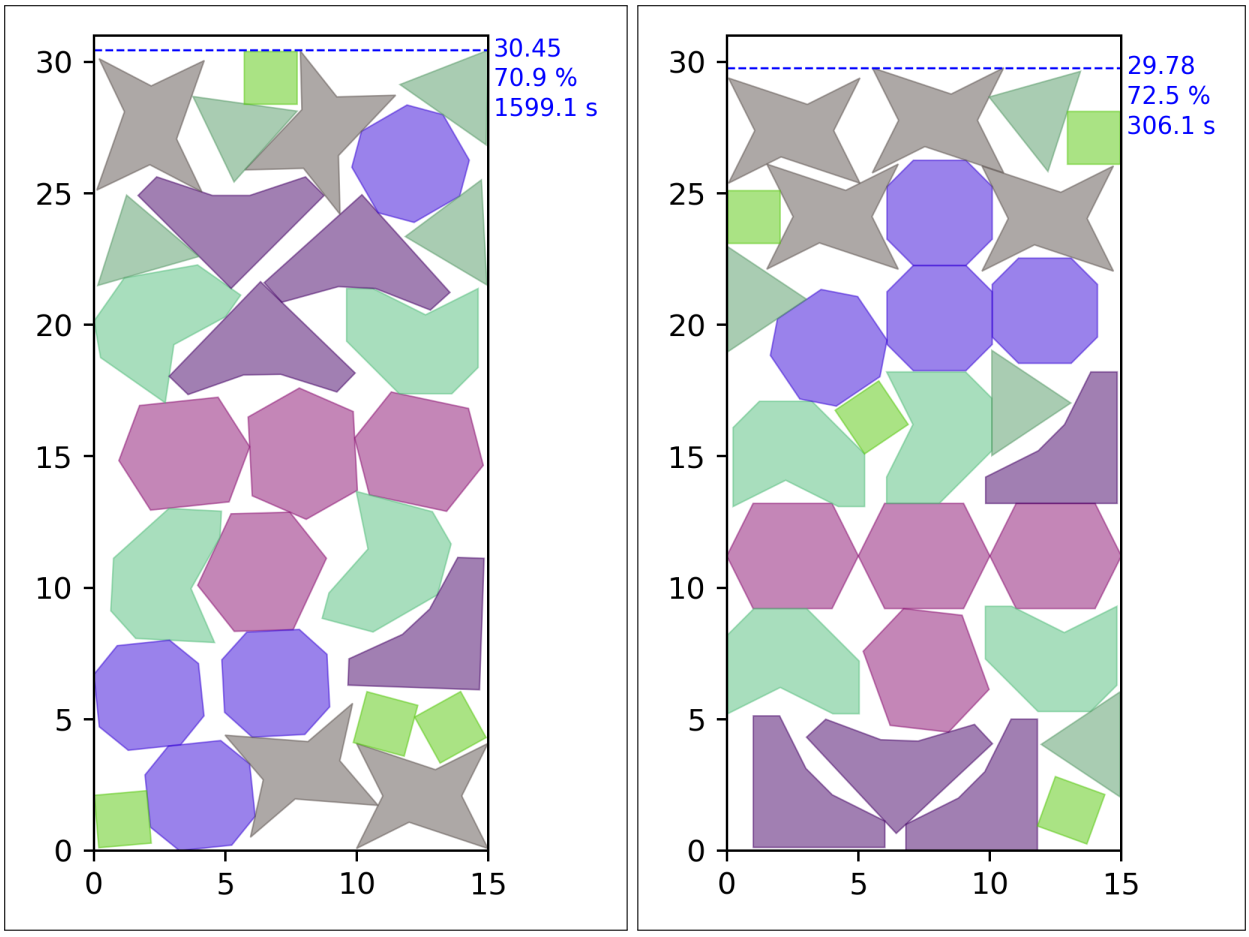
Anexo 3: Ilustração gráfica das soluções obtidas com o método proposto



(a) Algoritmo IGO_RS.

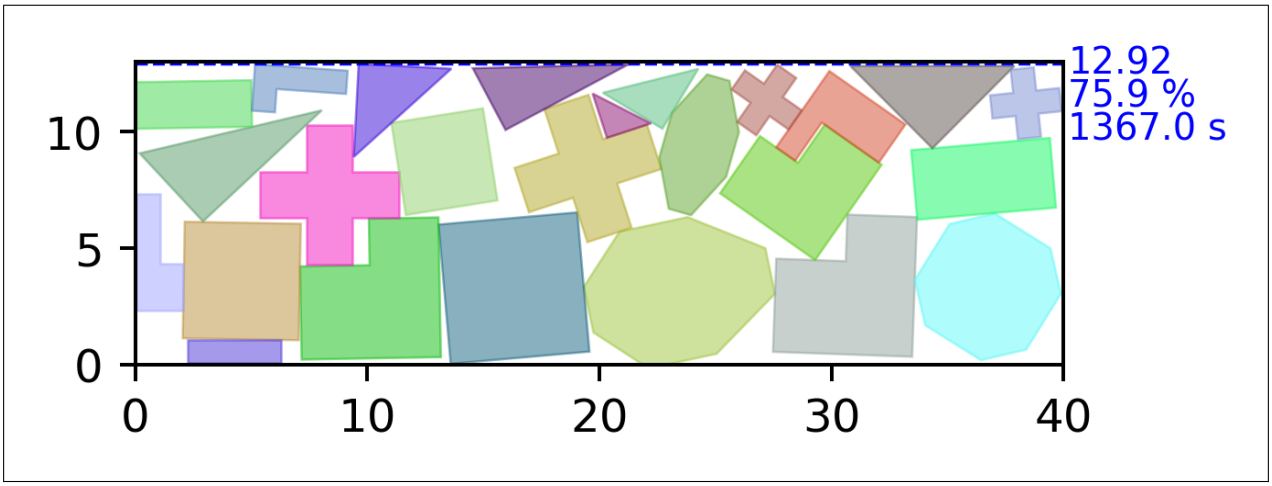
(b) Algoritmo IGO_RS_OP.

Figura 27: Soluções da instância *Albano* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria

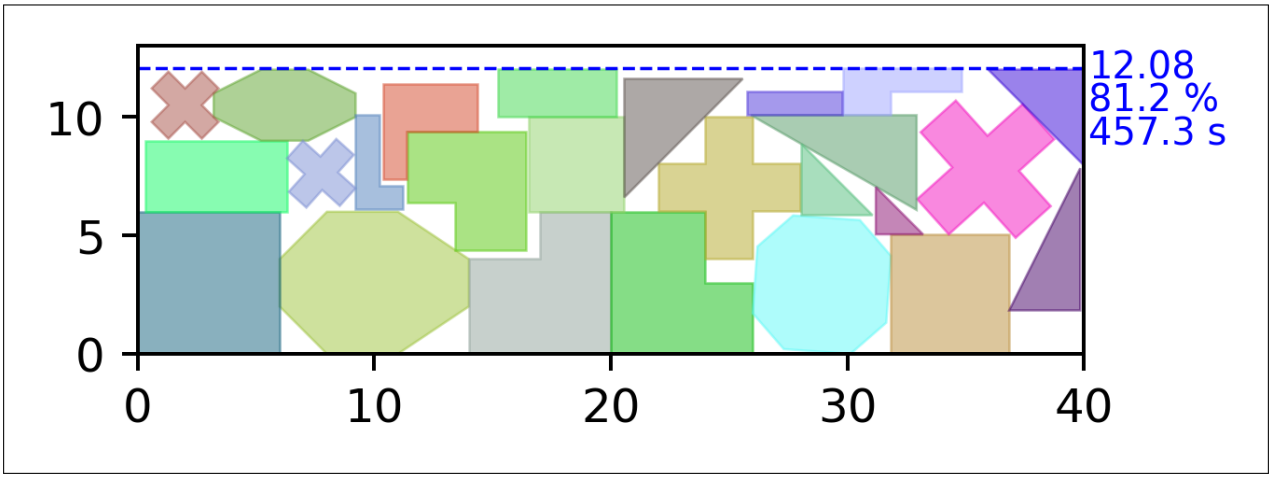


(a) Algoritmo IGO_RS. (b) Algoritmo IGO_RS_OP.

Figura 28: Soluções da instância *Blaz* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria

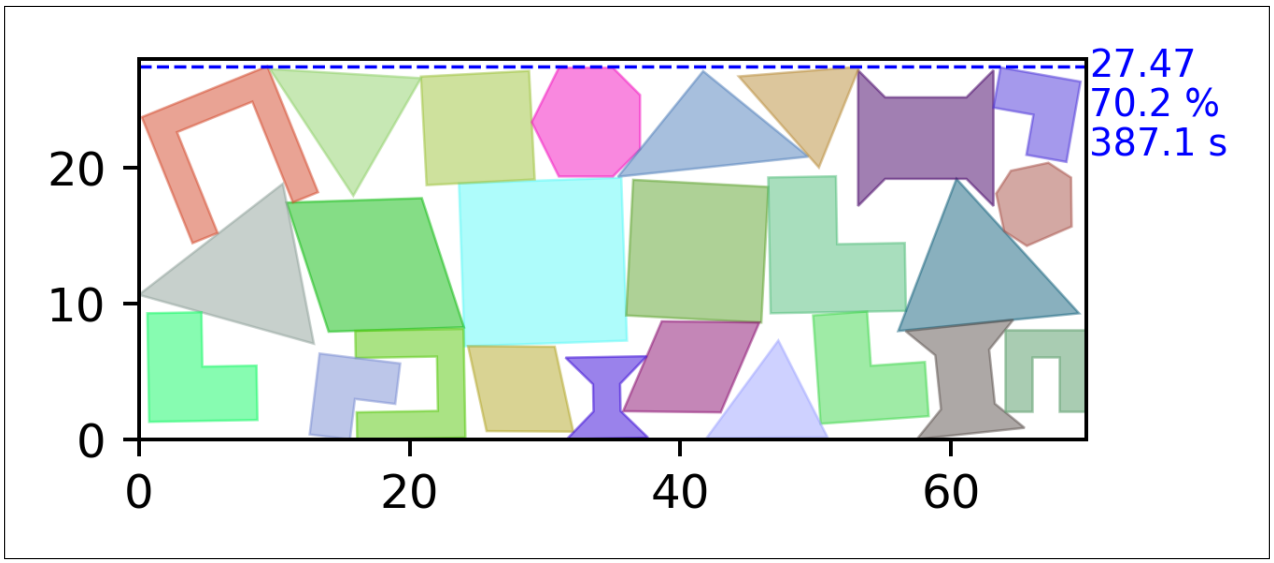


(a) Algoritmo IGO_RS.

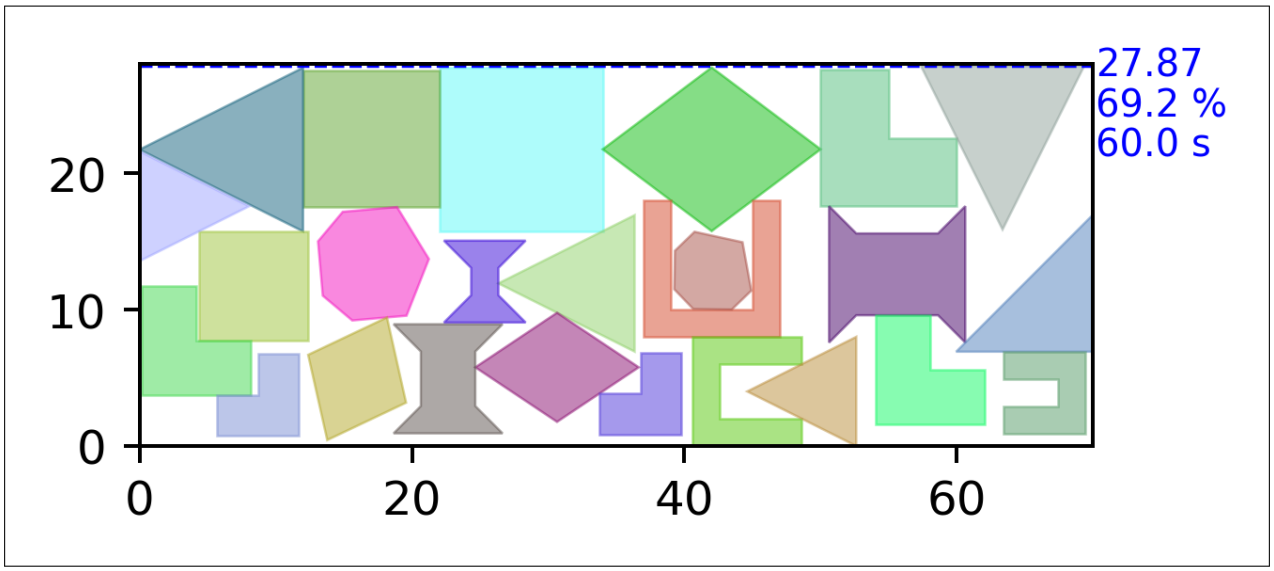


(b) Algoritmo IGO_RS_OP.

Figura 29: Soluções da instância *Jakobs* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria

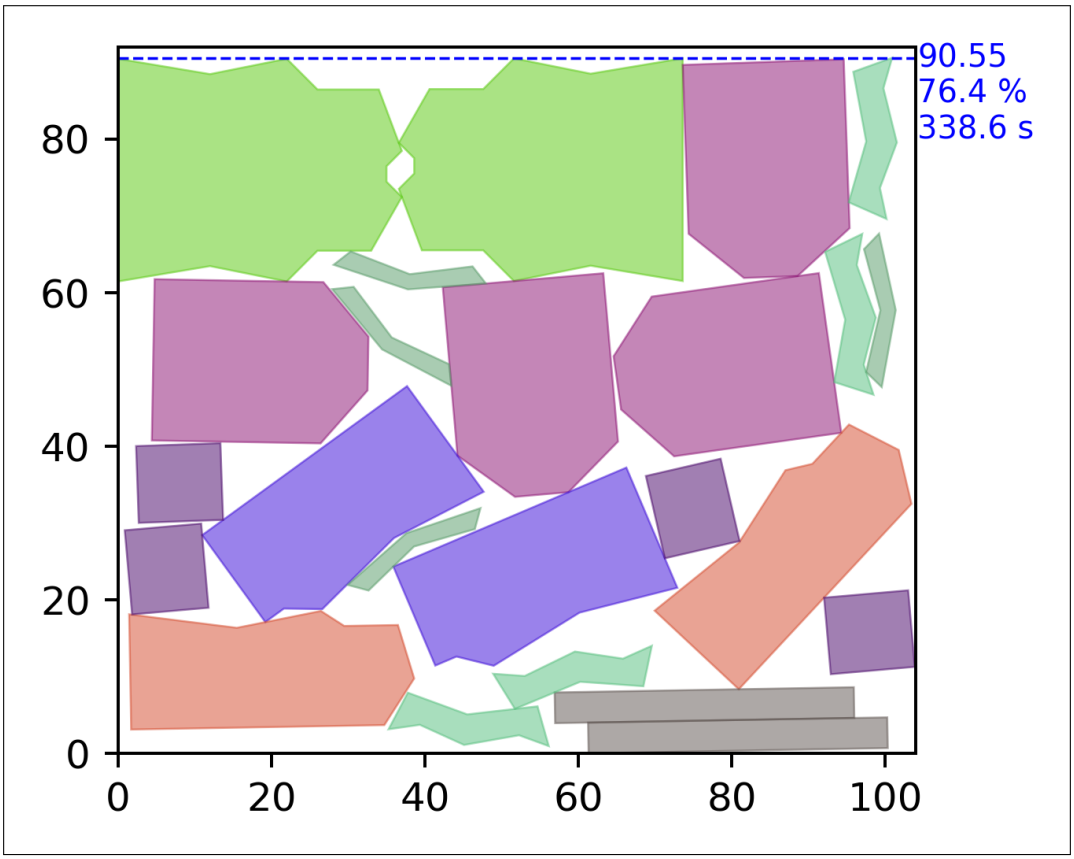


(a) Algoritmo IGO_RS.

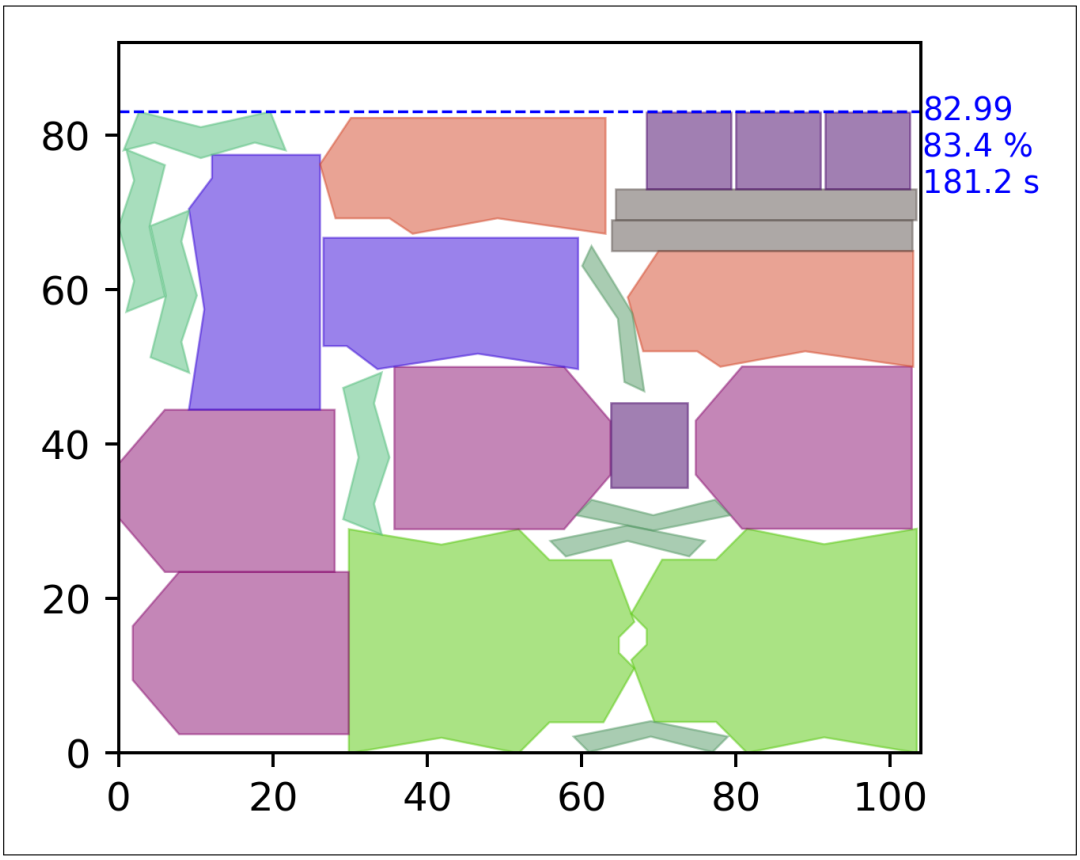


(b) Algoritmo IGO_RS_OP.

Figura 30: Soluções da instância *Jakobs2* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria

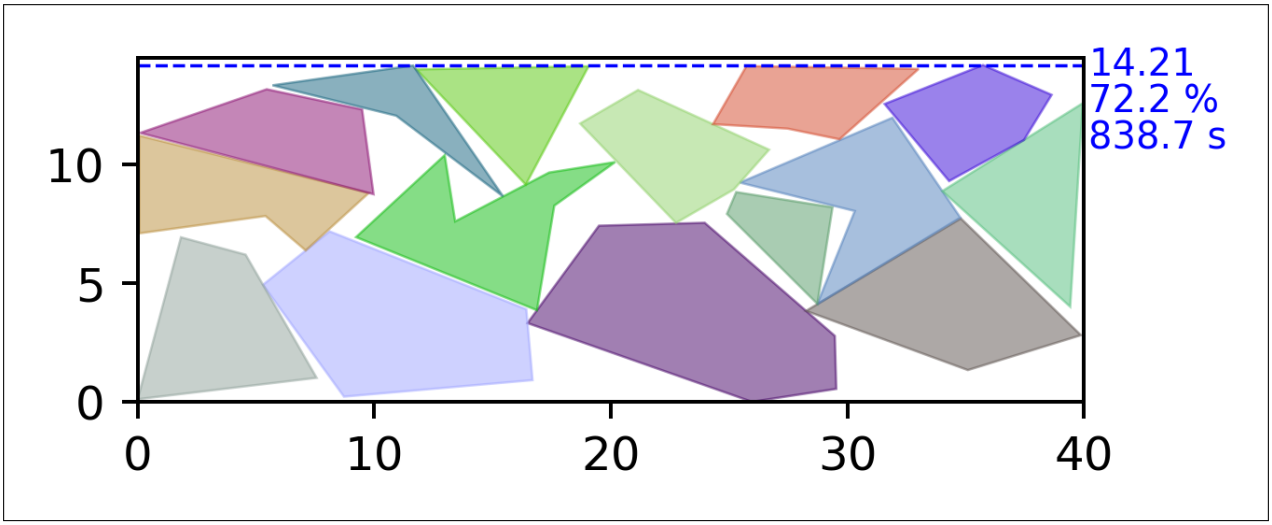


(a) Algoritmo IGO_RS.

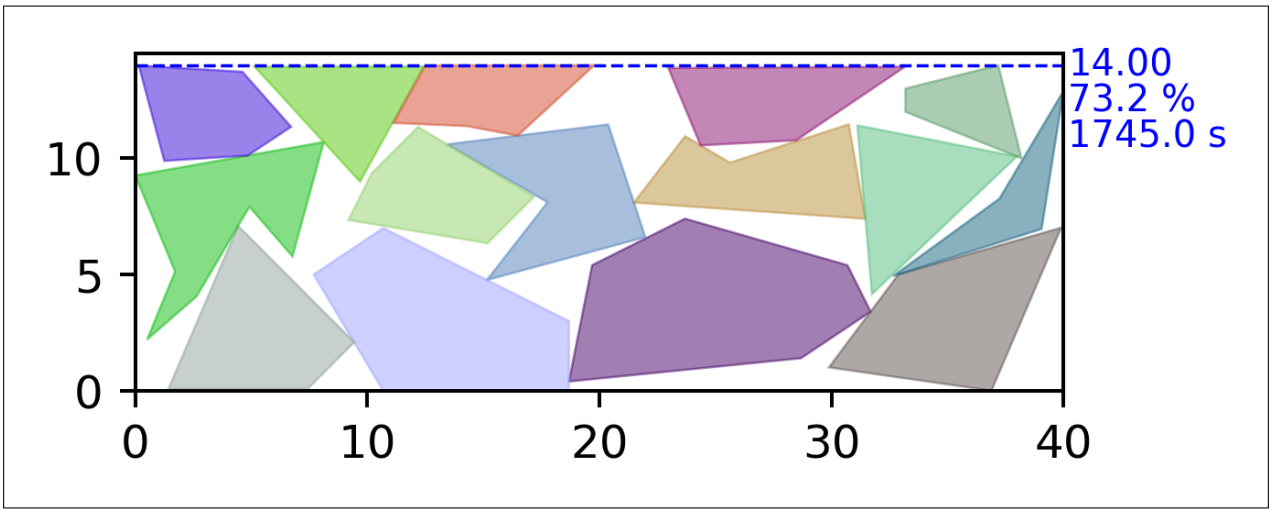


(b) Algoritmo IGO_RS_OP.

Figura 31: Soluções da instância *Marques* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria

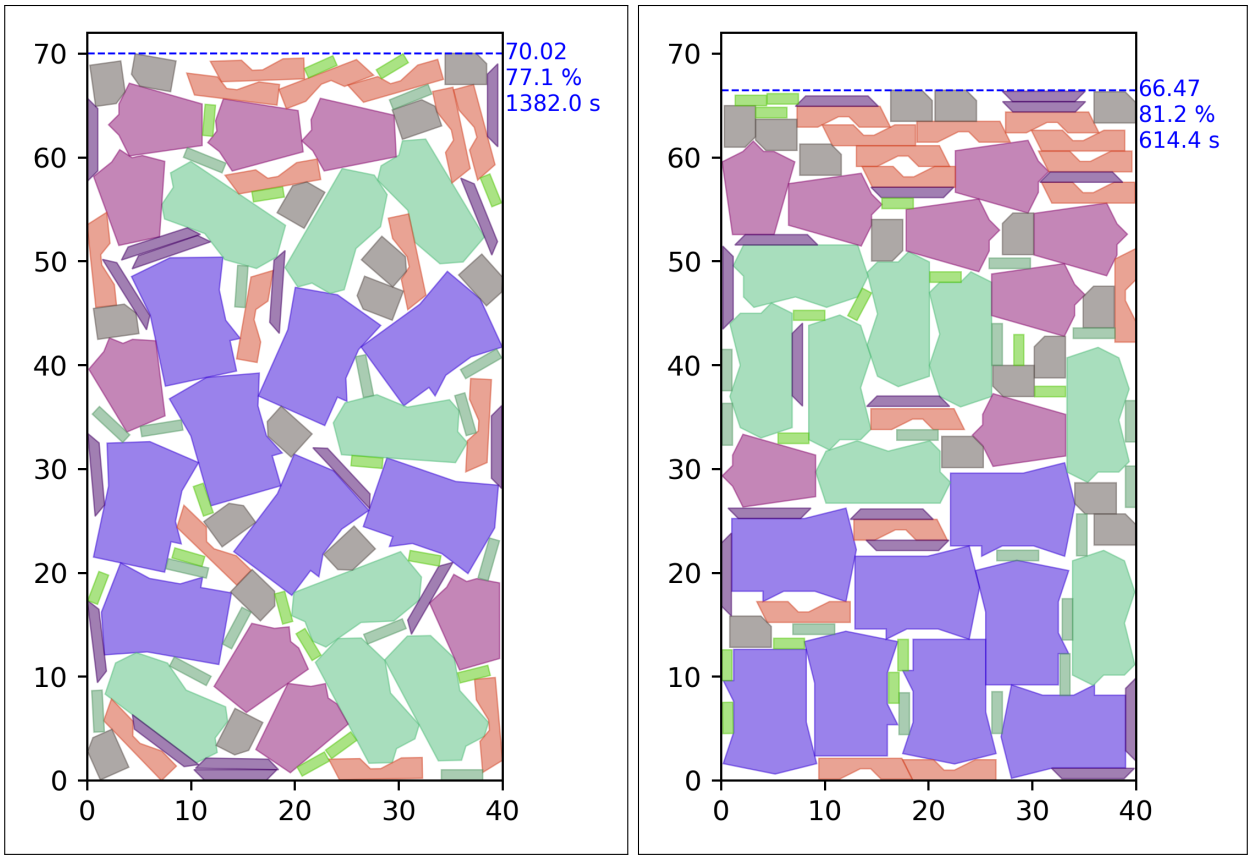


(a) Algoritmo IGO_RS.



(b) Algoritmo IGO_RS_OP.

Figura 32: Soluções da instância *Poly1a* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria



(a) Algoritmo IGO_RS. (b) Algoritmo IGO_RS_OP.

Figura 33: Soluções da instância *Shirts* obtidas pelos algoritmos IGO_RS e IGO_RS_OP. Fonte: Própria