UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

DEUSIANE KAYLANE MOREIRA MAIA

ALGORITMO GENÉTICO APLICADO AO PROBLEMA DE CORTE BIDIMENSIONAL E GUILHOTINADO EM PLACAS DE MADEIRA

RUSSAS

2026

DEUSIANE KAYLANE MOREIRA MAIA

ALGORITMO GENÉTICO APLICADO AO PROBLEMA DE CORTE BIDIMENSIONAL E GUILHOTINADO EM PLACAS DE MADEIRA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientador: Prof. Dr. Mayrton Dias de Queiroz

RUSSAS

2026

DEUSIANE KAYLANE MOREIRA MAIA

ALGORITMO GENÉTICO APLICADO AO PROBLEMA DE CORTE BIDIMENSIONAL E GUILHOTINADO EM PLACAS DE MADEIRA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 14/07/2026

BANCA EXAMINADORA

Prof. Dr. Mayrton Dias de Queiroz (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Ms. Hugo Nathan Barbosa Regis
Universidade Federal do Ceará (UFC)

Profª. Esp. Kátia Rocha de Almeida
Universidade Federal do Pernambuco (UFPE)

AGRADECIMENTOS

Primeiramente, eu agradeço a minha mãe, Karla, por todo o apoio, por me dar educação e sempre me incentivar em tudo o que me proponho na vida, principalmente a estudar, a senhora sempre será minha maior inspiração e força, te amo mãe. Aos meus irmãos, Klara e Wagner pelas risadas e momentos de descontração. Agradeço também ao meu avô, Sabino, pelas histórias e palavras de incentivo nas minhas visitas nas férias, à minha tia Acácia por todas as risadas, pelo apoio emocional, e por nossas viagens para esfriar a mente. Agradeço também ao resto de minha família, meu pai Deusimar, meus outros avós, Francisca e Jeová, minhas tias e meus tios, Márcia, Eriberto, Cícera, Elizabete e Maurício. Dedico meu TCC a vocês, que são tão especiais na minha vida.

Agradeço também a todos os amigos que fiz nessa jornada da universidade, vocês com certeza foram um dos pilares fundamentais nessa caminhada. Agradeço especialmente a: Sofia, por ser sempre a minha parceira de pesquisa, por sempre me escutar, aconselhar e propor novos desafios, Simires, por ser minha dupla de cadeiras, sempre paciente e calma, mas com os melhores comentários, Lince, por ser uma fonte de inspiração artística e sempre me alegrar com suas histórias, Daniel, por ser meu parceiro de academia e sempre me lembrar que sou uma pessoa capaz, e Mikael, por todos os comentários doces e caronas de fim de tarde, amo todos vocês. Agradeço também ao meu namorado, Enzo, pelo apoio, conselhos, carinho e palavras de afirmação, você é ótimo, te amo. Agradeço também a todos os outros amigos do grupo camaradas, João Lucas, Pedro, Thalles, Matheus, Douglas, Arthur e Hítalo, vocês são ótimos. Agradeço também as pessoas do laboratório LUDI, especialmente a Andreza, Ana Emilly, Kelly e Maria Julia.

Agradeço também aos amigos que fiz fora da universidade, especialmente a Yara, minha amizade mais antiga, minha prima e minha irmã de outra mãe, que acompanhou todas as minhas jornadas e acompanhará as futuras também, amo você. Aos meus amigos do ensino médio, Pedro e Athy, por todas as risadas, carinho, escuta sem julgamentos e conselhos, também amo vocês.

Agradeço a todos os professores do Campus de Russas, especialmente ao meu orientador, Dr. Mayrton Queiroz, pela dedicação, paciência e ensinamentos. Sua orientação impecável fez toda a diferença, aprendi muito durante essa jornada. À Dra. Anna Beatriz, expresso minha profunda gratidão pelas oportunidades concedidas; a senhora é uma verdadeira inspiração para mim. Estendo meus agradecimentos à Ms. Valéria, pela convivência gratificante

e pelas diversas orientações de artigo, onde sempre demonstrou uma capacidade incrível de nos orientar. Ao coordenador, Dr. Anderson Feitoza, agradeço pelo excelente trabalho e pela prontidão em sempre resolver os problemas dos alunos, e à Dra. Tatiane Figueiredo, por também ser uma grande fonte de inspiração. Agradeço também aos professores que marcaram outros momentos da minha vida, pois acredito que a educação é libertadora. De modo especial, aos professores Rômulo e Gutemberg, que me despertaram o interesse pela área de TI, a professora Marciana Sobreira, suas aulas sempre foram marcantes, os professores Theo e Alberto, por acreditarem na minha capacidade desde o ensino fundamental.

Agradeço também a todos os servidores técnico-administrativos e demais funcionários da UFC – Campus de Russas, que com dedicação e trabalho diário tornam o ambiente acadêmico acolhedor e viabilizam o funcionamento da nossa instituição.

“O que verdadeiramente somos é aquilo que o impossível cria em nós.”

(Clarisse Lispector)

RESUMO

As indústrias da modernidade buscam otimizar continuamente os seus processos de produção, seja diminuindo o desperdício de insumos e tempo de fabricação, seja aumentando a eficiência e a quantidade de produtos produzidos. Na indústria de móveis não é diferente, pois de maneira recorrente enfrenta níveis de desperdício de matéria-prima, principalmente no que tange aos cortes de placas de madeira. Nesse sentido, este trabalho tem como objetivo identificar uma estratégia capaz de encontrar uma solução para a realização de cortes de peças em placas de madeira, para maximizar o seu aproveitamento e em tempo hábil. O presente trabalho aborda os problemas de corte e empacotamento bidimensionais, uma vez que consideram-se as dimensões de largura e altura das placas de madeira, com cortes guilhotinados em peças retangulares que apresentam padrões geométricos e que não sofrem rotações devido ao design do móvel. Inicialmente, é realizada uma Revisão Sistemática da Literatura com o objetivo de identificar e analisar as abordagens existentes relacionadas a essa temática. Posteriormente, são implementadas e avaliadas seis configurações de um algoritmo genético, obtidas a partir das combinações dos operadores de Seleção por Roleta e Torneio, e dos operadores de Cruzamento de Um Ponto, Dois Pontos e Pai Único, visando obter soluções de boa qualidade em tempo computacional viável. Dentre as seis configurações avaliadas, o Algoritmo 6, composto pelos operadores de Seleção por Torneio e Cruzamento Pai Único, apresentou a melhor escalabilidade e o melhor desempenho geral, em termos de tempo de execução e aproveitamento da placa. Para as instâncias de pequeno porte, como a m0_i1 (10 peças), os Algoritmos 4, 5 e 6 apresentaram uma redução de aproximadamente 50% no tempo de execução em relação aos Algoritmos 1, 2 e 3, com o Algoritmo 6 registrando o menor tempo de execução, de aproximadamente 0.1136 segundos. Já na instância mais complexa (m5_i40, com 4240 peças), o Algoritmo 6 executou em aproximadamente 25.0 segundos, mantendo o melhor desempenho entre as configurações avaliadas. Além disso, o Algoritmo 6 demonstrou a maior estabilidade, com a amplitude interquartil (IQR) mais compacta entre os *boxplots* analisados. Dessa forma, o Algoritmo 6 consolida-se como a configuração proposta neste trabalho mais adequada para cenários industriais reais, onde o elevado volume de peças exige respostas rápidas e confiáveis.

Palavras-chave: problema de corte; otimização de corte em chapas de madeira; algoritmo genético; corte bidimensional guilhotinado.

ABSTRACT

Modern industries seek to continuously optimize their production processes, whether by reducing waste of raw materials and manufacturing time or by increasing efficiency and the quantity of products produced. The furniture industry is no exception, as it repeatedly faces high levels of raw material waste, particularly when cutting wooden panels. In this regard, this study aims to identify a strategy capable of finding a solution for cutting pieces from wooden panels to maximize material utilization in a timely manner. This study addresses two-dimensional cutting and nesting problems, as it considers the width and height dimensions of the wood panels, with straight cuts into rectangular pieces that follow geometric patterns and do not undergo rotation due to the furniture's design. Initially, a systematic literature review is conducted to identify and analyze existing approaches related to this topic. Subsequently, six configurations of a genetic algorithm are implemented and evaluated, obtained from combinations of the Roulette Wheel and Tournament selection operators, and the One-Point, Two-Point, and Single-Parent crossover operators, with the aim of obtaining high-quality solutions within a feasible computational time. Among the six configurations evaluated, Algorithm 6, consisting of the Tournament Selection and Single-Parent Crossover operators, demonstrated the best scalability and overall performance in terms of execution time and GPU utilization. For small-scale instances, such as m0_i1 (10 pieces), Algorithms 4, 5, and 6 showed an approximately 50% reduction in execution time compared to Algorithms 1, 2, and 3, with Algorithm 6 recording the shortest execution time of approximately 0.1136 seconds. In the most complex instance (m5_i40, with 4,240 pieces), Algorithm 6 ran in approximately 25.0 seconds, maintaining the best performance among the evaluated configurations. Furthermore, Algorithm 6 demonstrated the greatest stability, with the narrowest interquartile range (IQR) among the *boxplots* analyzed. Thus, Algorithm 6 emerges as the configuration proposed in this study that is best suited for real-world industrial scenarios, where the high volume of parts demands fast and reliable responses.

Keywords: cutting problem; cutting optimization in wood sheets; genetic algorithm; two-dimensional guillotine cutting.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo genético
BRKGA	Algoritmo Genético de Chaves Aleatórias Viciadas
FAO	*Food and Agriculture Organization*
FFDH	*First Fit Decreasing Height*
GRASP	*Greedy Randomized Adaptive Search Procedure*
IQR	*Intervalo Interquartil*
MPI	*Message Passing Interface*
PCE	Problemas de Corte e Empacotamento
RSL	Revisão Sistemática da Literatura

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo será abordado a contextualização do trabalho, a justificativa, os objetivos gerais e específicos e a organização do trabalho.

1.1 Contextualização

Na indústria atual, em que a produção segue padrões para maximização dos lucros e minimização dos custos, soluções algorítmicas que possuem esses objetivos tornam-se cada vez mais relevantes. Nesse sentido, surgem os chamados Problemas de Corte e Empacotamento (PCE), no qual há um grupo de peças que serão cortadas, a fim de cumprir as especificações de tamanho e quantidade pré-determinadas (Gramani, 1997). Os problemas de corte envolvem cortar peças, sejam elas de madeira, vidro ou qualquer outro material, maiores em itens menores, buscando reduzir perdas e atender demandas específicas em diversos contextos industriais (Vianna, 2000).

Considerando esses aspectos, diversas indústrias precisam realizar cortes em seus materiais para a produção de suas peças, o que explica a atenção dada aos PCE's. Essa relevância decorre principalmente de sua ampla aplicabilidade e impacto direto na eficiência produtiva, já que tais problemas aparecem de forma recorrente em setores como aço, papel, vidro e têxteis (Bischoff; Wäscher, 1995).

Para a resolução dos PCE's, diferentes abordagens algorítmicas têm sido empregadas, incluindo métodos exatos e heurísticos. As abordagens exatas buscam obter a solução ótima diante da modelagem matemática definida, isto é, o melhor resultado possível em termos de aproveitamento do material, contudo, à medida que a instância aumenta seu tamanho, ou novas restrições são consideradas na modelagem, essas técnicas tendem a demandar um tempo computacional elevado, em razão do grande número de combinações possíveis. Por outro lado, as heurísticas apresentam menor custo computacional e maior viabilidade prática em cenários reais, embora não garantam a obtenção da solução ótima, oferecendo, em contrapartida, soluções de boa qualidade em tempo reduzido (Oliveira, 2018).

Ademais, os PCE's possuem diversas variantes, conforme descrito Silva (2011), como a dimensionalidade do corte, os padrões geométricos, o maquinário utilizado que realiza diferentes tipos de corte e restrições impostas ao processo produtivo. Dessa forma, destacam-se: os problemas de corte unidimensionais, quando considera-se uma única dimensão para

realizar os cortes, problemas de corte bidimensionais, que envolve o corte de peças planas considerando apenas altura e largura. Existem ainda extensões para o corte tridimensional, aplicadas, por exemplo, à indústria de contêineres. Além disso, as peças a serem cortadas também podem se diferenciar geometricamente, sendo divididas em regulares, com peças no formato de quadrados, retângulos, losangos e círculos. Já as peças irregulares, possuem partes retilíneas e curvilíneas, necessitando de um equipamento que realize o corte de forma adequada sem danificá-las (Wäscher; Haußner; Schumann, 2007).

Diante deste cenário, surge o desafio de encontrar algoritmos de otimização que determinem uma sequência de cortes nas placas de madeira, de forma que seja levado em consideração a maximização do saldo da placa de madeira, quantidade de peças cortadas e tempo de processamento. É importante destacar que existem trabalhos na literatura que adotaram heurísticas, como: *Bottom-Left* (Sepulveda, 2013), o *First Fit Decreasing Height* (FFDH) (Muzulon; Verga, 2019), Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) (Queirós, 2019), entre outras, no entanto, é importante a investigação de algoritmos com intuito de encontrar melhores sequências de cortes, visto que existem diversas variações do problema, quanto com relação às peças: quantidade, formato, orientação. Como também em relação ao tipo do corte e ao equipamento adotado.

1.2 Justificativa

A indústria moveleira apresenta forte relevância econômica no Brasil, e caracteriza-se por uma produção em larga escala, o que implica a elevada demanda por matéria-prima, especialmente placas e chapas de madeira. Segundo o Relatório Setorial da Indústria de Móveis no Brasil 2024 (Móveis, 2024), o setor produziu aproximadamente 439.9 milhões de peças de móveis e colchões, sendo que a maior parte dessa produção esteve concentrada em móveis de madeira maciça, painéis e chapas, que totalizaram cerca de 319 milhões de peças no período. Diante desse cenário, o desperdício de matéria-prima ao longo do processo produtivo torna-se um fator crítico, uma vez que impacta diretamente os custos de produção, o consumo de insumos naturais e a competitividade das empresas.

Do ponto de vista ambiental, a produção de móveis gera impactos diretos no meio ambiente. A indústria moveleira, por utilizar a madeira como matéria-prima principal, gera resíduos como serragem, cavacos, lodo de estação de tratamento de efluentes e resíduos de tintas e solventes em diversas etapas do processo produtivo (Macedo *et al.*, 2013). Além disso, a

demanda por madeira cresce a cada ano, com projeções da *Food and Agriculture Organization* (FAO) indicam que o consumo de madeira redonda pode aumentar em até 49% até o ano de 2050, impulsionado principalmente pela demanda por madeira de uso industrial (FAO, 2022). Esse dado reforça a urgência de práticas industriais que utilizem a matéria-prima de forma eficiente, evitando o desperdício, reduzindo a necessidade de extração, e consequentemente, reduzindo a pressão acerca da extração de recursos naturais.

1.3 Objetivos

Nesse contexto,o objetivo deste trabalho consiste em identificar um algoritmo capaz de encontrar uma sequência de cortes guilhotinados em placas de madeira com peças orientadas, com intuito de maximizar o saldo, dessa forma será utilizado o Algoritmo genético (AG), bem como a combinação dos seus operadores.

1.3.1 Objetivos específicos

A seguir, é possível observar os objetivos específicos que este trabalho almeja alcançar:

- Identificar os métodos existentes na literatura para resolução de problemas de corte, com ênfase em abordagens exatas, heurísticas e metaheurísticas.
- Obter um algoritmo genético alinhado ao problema, definindo os operadores de seleção, cruzamento e mutação.
- Obter os experimentos computacionais com diferentes conjuntos de instâncias e parâmetros, a fim de verificar o desempenho do algoritmo em termos de aproveitamento de material, tempo de execução e estabilidade.
- Identificar, dentre as 6 configurações do AG, qual possui melhor desempenho, em termos de aproveitamento de material, tempo de execução e estabilidade.

1.4 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- **Capítulo 1:** Aborda a introdução, objetivos gerais, objetivos específicos e a organização do trabalho.
- **Capítulo 2:** Explica o contexto do trabalho, com assuntos que são relevantes para o estudo,

como do que se tratam os problemas de corte e empacotamento, algoritmo utilizado e a relevância do trabalho tanto para o meio ambiente quanto para a literatura.

- **Capítulo 3:** Trata da literatura atual do que tange os problemas de corte, a partir da aplicação do método da Revisão Sistemática da Literatura.
- **Capítulo 4:** Envolve a descrição das etapas desenvolvidas durante a realização deste trabalho.
- **Capítulo 5:** Diz respeito aos resultados que este trabalho almeja alcançar a partir da solução produzida.
- **Capítulo 6:** Aborda as conclusões preliminares que o atual trabalho alcançou.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, será apresentada a base teórica necessária para a compreensão e o desenvolvimento deste trabalho, contendo: a descrição dos problemas de corte bidimensionais aplicados a peças de madeira, os conceitos dos problemas de otimização, e detalha-se o funcionamento do Algoritmo Genético.

2.1 Problemas de corte bidimensionais em peças de madeira

Os denominados Problemas de Corte e Empacotamento (PCEs) não constituem um tema recente na literatura. Pelo contrário, vem sendo estudado desde a década de 1960, a partir dos trabalhos de Gilmore and Gomory (1961), que estabeleceram as bases para a otimização de problemas de corte unidimensionais. Os PCEs referem-se ao processo de divisão de uma matéria-prima de maiores dimensões em peças menores (Viana *et al.*, 2021), com o objetivo de minimizar perdas e atender a uma demanda específica de tamanhos e quantidades.

Esses problemas são relevantes em diversos setores da indústria moderna, como as indústrias têxtil, vidreira, de celulose e papel e, especialmente, a indústria moveleira. Nesse contexto, a indústria moveleira representa uma instância clássica dos Problemas de Corte, cujo principal objetivo é minimizar o desperdício de material ou maximizar o aproveitamento das placas de madeira. Existem diferentes tipos de cortes utilizados na indústria, entre os quais se destaca o corte guilhotinado, que consiste em dividir um material retangular em dois novos retângulos por meio de um corte realizado de uma extremidade à outra da chapa, sem interrupções ou mudanças de direção (Silveira; Morabito, 2002). Por outro lado, os cortes não guilhotinados exigem a interrupção do processo de corte, uma vez que sua execução demanda a rotação da chapa ou do equipamento de corte, o que implica maior complexidade operacional (Hoffmann *et al.*, 2015).

Para além disso, outra restrição que as peças podem apresentar após o corte diz respeito à sua geometria. Para Wäscher, Haußner and Schumann (2007), as peças podem possuir formato regular, quando as formas são bem definidas (retângulos, quadrados, cilindros, círculos, entre outros), ou irregular, aquelas que não seguem um padrão geométrico definido, como um corte para um calçado.

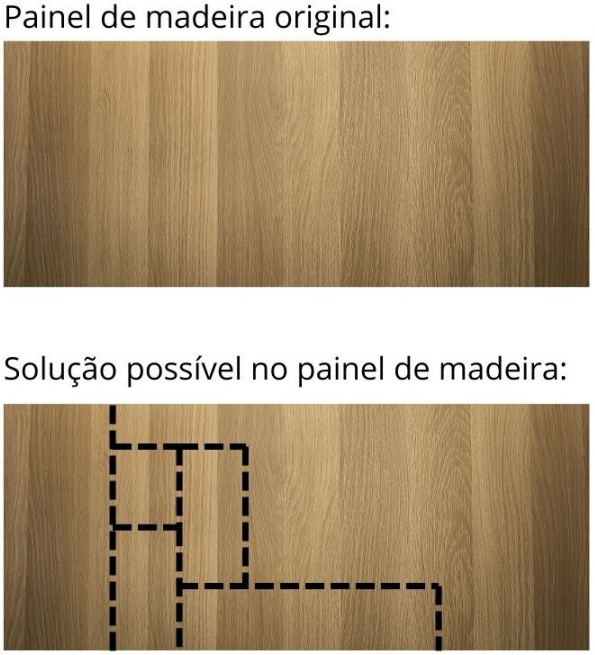
Outro aspecto relevante diz respeito a orientação das peças, isto é, algumas placas de madeira possuem texturas, o que resulta na não rotação da peça durante o corte, já que o padrão

precisa ser respeitado para não afetar a construção dos itens. Nesse caso, admite-se então que a peça possui uma orientação fixa e não deve ser rotacionada (Martins, 2010).

Por fim, outro fator importante são as dimensões que os cortes da peça devem assumir. No trabalho Wäscher, Haußner and Schumann (2007), há descrição de três tipos de dimensões que os itens podem assumir: unidimensional, no qual apenas uma dimensão é levada em consideração, como em barras de aço, papel ou o Problema da Mochila; bidimensional, em que duas dimensões devem ser consideradas simultaneamente, como em chapas de metais, tecidos, ou placas de madeira; e tridimensionais, onde são consideradas as dimensões de largura, altura e comprimento, como em contêineres ou caixas.

Na Figura 1, é demonstrada uma placa de madeira com textura, ou seja, que possui orientação e no momento do corte a peça não poderá ser rotacionada, tendo em vista os padrões gráficos da peça.

Figura 1 – Exemplo de painel de madeira sujeito a cortes.



Fonte: Elaborada pela autora (2025).

2.2 Problemas de otimização

Problemas de otimização são recorrentes na literatura e têm como objetivo identificar a melhor solução, seja de maximização ou de minimização, considerando um conjunto de restrições previamente estabelecidas. Para isso, utilizam-se modelos matemáticos que permitem

a representação do problema e a busca por soluções ótimas. Os problemas de maximização visam obter o maior valor possível para uma função objetivo, enquanto problemas de minimização buscam encontrar o menor valor com base na função objetivo (Oliveira; Lopes, 2022).

Na área da Computação, problemas de otimização são amplamente estudados e aplicados, contando com diversas técnicas para sua resolução. Entre essas técnicas, destacam-se os algoritmos exatos, que têm como finalidade encontrar a solução ótima do problema. No entanto, para instâncias maiores e com um número elevado de restrições, esses algoritmos produzem soluções com tempo computacional inviável na prática. Diante dessa limitação, surgem as abordagens heurísticas, que buscam soluções aproximadas à ótima, priorizando a obtenção de resultados de boa qualidade em um tempo de processamento reduzido (Silva *et al.*, 2024).

2.2.1 Algoritmo genético

Algoritmos Genéticos são algoritmos heurísticos de otimização baseados na teoria de seleção natural, proposta por Charles Darwin, na qual indivíduos mais aptos possuem maior probabilidade de sobreviver e gerar descendentes (Linden, 2008). Na computação, esses princípios são adaptados para a construção de algoritmos capazes de encontrar boas soluções para problemas complexos de otimização, por meio da evolução de uma população de soluções candidatas.

O AG funciona a partir de soluções, cada solução do problema é representada por um indivíduo, também denominado cromossomo, e o conjunto desses indivíduos forma uma população. Ao longo de sucessivas gerações, essa população evolui por meio de operadores genéticos, de modo que indivíduos mais aptos tendem a predominar (Pacheco *et al.*, 1999).

Primeiramente, no AG há a criação da população inicial, que normalmente é formada por um conjunto de indivíduos, que podem ser gerados de forma aleatória, garantindo diversidade genética no início do processo evolutivo (Pacheco *et al.*, 1999). Logo depois, cada indivíduo é avaliado pela função *fitness*. De posse do valor de avaliação do indivíduo, é iniciada a seleção de pais que vão participar da próxima etapa de reprodução, em que a seleção dos indivíduos mais aptos para a geração de cromossomos filhos é feita a partir da aplicação dos operadores genéticos de *crossover* e mutação. O operador *crossover* simula a troca de genes entre dois pais para a geração de novos descendentes (Pinto; Martarelli; Nagano, 2022). Logo após o *crossover* ser aplicado, o operador de mutação entra em ação, podendo alterar os genes, com o objetivo de

expandir a diversidade da população e explorar outras regiões do espaço de busca.

No Algoritmo 1, é possível observar o pseudocódigo de um algoritmo genético básico, adaptado de (Queiroz *et al.*, 2019).

Algoritmo 1: ALGORITMO GENÉTICO

```
   Entrada: tamanho, gerações, mutação
   Saída: Melhor solução encontrada
 1 início
 2     população = criarPopulação(tamanho)
 3     para cont de 0 até gerações faça
 4         avaliar(população)
 5         para i de 0 até tamanho faça
 6             pai1 = selecionar()
 7             pai2 = selecionar()
 8             filho = reproduzir( pai1, pai2)
 9             novoFilho = mutar( filho, taxa)
10             novaPopulação[i] = novoFilho
11         fim
12         moduloPopulação( população, novaPopulação)
13         melhorFilho = selecionarMelhor( população)
14     fim
15     retorna melhorFilho
16 fim
```

Fonte: (Queiroz *et al.*, 2019).

Na linha 2, observa-se a criação da população inicial a partir da definição de um tamanho da população anteriormente definido, passado neste caso por parâmetro. Cada indivíduo dessa população representa uma solução candidata ao problema. Logo depois inicia-se um laço de repetição que vai de zero até a quantidade de gerações, servindo como critério de parada do AG. Neste primeiro laço de repetição, é realizada a avaliação da qualidade das populações geradas, por meio da função de avaliação presente na linha 4. Posteriormente, entre as linhas 6 e 10, têm-se os operadores de cruzamento, seleção e mutação, a fim de gerar variabilidade nas soluções. Esses operadores são executados em um laço interno que se repete até que uma nova população seja completa. Nas linhas 12 e 13, ocorre a atualização da população, substituindo a população antiga pela nova população gerada e também ocorre a seleção do melhor filho da população atual. Por fim, após o número de gerações definidas serem alcançadas, o algoritmo irá retornar o melhor filho gerado, isto é, a solução mais bem avaliada.

3 REVISÃO DA LITERATURA

Neste capítulo, apresenta-se a metodologia utilizada para a realização da Revisão Sistemática da Literatura (RSL), será abordado: o contexto da pesquisa, o planejamento para a execução, a *string* de busca usada, as bases de dados, os critérios de inclusão e exclusão, as questões de pesquisa, a condução e os resultados obtidos.

3.1 Contextualização

Uma Revisão Sistemática da Literatura é um método formal, que segue protocolos específicos de busca, com intuito de identificar, analisar e avaliar trabalhos relacionados a um determinado contexto, além disso, é passível de réplica por terceiros. Para Kitchenham and Charters (2007), a revisão deve empregar estratégias de busca abrangentes e utilizar critérios explícitos de inclusão e exclusão para reduzir vieses e garantir a segurança e replicabilidade dos resultados, de tal forma que os autores organizam a revisão da literatura em três fases, sendo elas: planejamento, condução e resultados.

3.2 Planejamento

A revisão sistemática da literatura, ao contrário de outros métodos de revisão informais, segue processos bem definidos, que precisam ser seguidos de forma rigorosa para que os resultados sejam confiáveis e replicáveis. Tendo em vista esses aspectos, primeiramente foi estabelecida uma *string* de busca, as bibliotecas digitais para a seleção dos artigos, os critérios de inclusão e exclusão foram determinados com base no protocolo estipulado por Kitchenham and Charters (2007) e as questões de pesquisa foram elaboradas.

3.2.1 Definição da String de busca

A *string* de busca é o meio no qual se obtêm os trabalhos relacionados ao contexto determinado da pesquisa. Logo, os termos relacionados foram determinados com base nos PCE's em madeira. A *string* de busca utilizada neste trabalho foi:

"problema de corte bidimensional" AND "madeira"

3.2.2 Definição das bases de dados

Logo após a definição da *string* de busca, foram definidas as bases de pesquisa para os trabalhos serem selecionados, levando em consideração bases que possuem trabalhos relacionados na área da computação, conforme mostra o Quadro 1.

Quadro 1: Bases de dados utilizadas na pesquisa.

Base	***Link***
Google Acadêmico	https://scholar.google.com
Periódicos CAPES	https://periodicos.capes.gov.br
Sol SBC	https://sol.sbc.org.br

Fonte: Elaborado pela autora (2025).

3.2.3 Definição dos critérios de inclusão e exclusão

Os critérios de inclusão e exclusão foram definidos para que somente os trabalhos relacionados às questões de pesquisa fossem selecionados.

Critérios de inclusão:

1. Estudos completos publicados em revistas ou conferências sobre o Problema de Corte Bidimensional em Madeira;
2. Estudos experimentais com o objetivo de apresentar conceitos para o entendimento da área;
3. Disponíveis para *download* e ter sido publicado no período de 2010 a 2025.

Critérios de exclusão:

1. Estudos que não estejam relacionados ao Problema de Corte Bidimensional em Madeira;
2. Estudos que não respondem a nenhuma das questões de pesquisa;
3. Artigos duplicados, ou seja, aqueles encontrados em mais de uma das bases selecionadas;
4. Artigos que não passaram pelo critério de avaliação das conferências ou revistas.

3.2.4 Definição das questões de pesquisa

Uma parte fundamental para uma RSL é a elaboração das questões de pesquisa, as quais são respondidas durante a etapa de condução. Tais questões buscam averiguar o tipo de abordagem que o trabalho trata e o tipo de algoritmo, além dos tipos de cortes que são tratados nos trabalhos, qual o formato das peças e se leva em consideração a orientação da madeira. A seguir estão as questões de pesquisa desenvolvidas para esta pesquisa:

QP1 – Qual a abordagem adotada no trabalho?

☐ Exata

☐ Heurística

☐ Simulação

☐ Revisão da literatura

QP2 – Qual(is) o(s) algoritmo(s) escolhido(s) no trabalho?

☐ Algoritmo Genético

☐ GRASP

☐ BRKGA

☐ Colônia de Formigas

☐ *Simulated Annealing*

☐ Não se aplica

☐ Outro

QP3 – Qual o tipo de corte?

☐ Guilhotinado

☐ Não Guilhotinado

QP4 – Qual o formato das peças?

☐ Retangular

☐ Formato em L

☐ Irregular

☐ Outro

QP5 – Considera a orientação das peças?

☐ Sim

☐ Não

3.3 Condução

Durante a condução da Revisão Sistemática da Literatura, o que foi definido no planejamento foi executado, então a *string* de busca foi adaptada para cada base utilizada, os artigos foram baixados e cada trabalho foi analisado para verificar se adequava aos critérios de inclusão estabelecidos, caso contrário, era rejeitado. Na Tabela 1 possível visualizar a quantidade inicial de artigos. Ao final, os artigos aceitos foram lidos para que as questões de pesquisa fossem respondidas.

Tabela 1: Quantidade de artigos obtidos durante a busca, *download* e aplicação dos critérios de inclusão e exclusão.

Bases	Busca	*Download*	Critérios
Google acadêmico	73	58	19
Periódicos da capes	1	1	1
Sol SBC	1	1	1
Total	**75**	**60**	**21**

Fonte: Elaborada pela autora (2025).

3.4 Resultados

Em síntese, os 21 trabalhos foram aceitos, como é possível observar na Tabela 1. Estes trabalhos foram analisados para responder às cinco questões de pesquisa.

Ao examinar a Figura 2 é possível observar que 16 trabalhos utilizam abordagens heurísticas, cerca de 77,27%, dentre estes: o trabalho de Sepulveda (2013), na qual propõe uma abordagem para resolver o PCE com e sem rotação de peças, combinando codificação binária, heurística *Bottom-Left* e *Simulated Annealing*; Queirós (2019) utiliza a heurística BRKGA, com cenários gerados por distribuições Beta, seu trabalho demonstra eficiência computacional e analisa o impacto da incerteza nos padrões de corte, porém não leva em consideração se há orientações nas peças; Candido (2011) discute uma metodologia para a geração de padrões de corte bidimensionais guilhotinados, com e sem estágios e com possibilidade de rotação dos itens, combinando algoritmos genéticos para a seleção e agrupamento dos itens com heurísticas de encaixe para definir o arranjo geométrico dos cortes.

Em Rodrigues and Amaral (2016), os autores tratam em sua pesquisa o corte não-guilhotinado em peças retangulares, a partir da meta-heurística *Greedy Randomized Adaptive Search Procedure* (GRASP), baseada na junção de estratégias para fases construtivas e de busca local; Martinez (2014) combina diversos algoritmos para solucionar o problema, como a meta-heurística GRASP, o AG e o Busca em Vizinhança Variável, para tratar peças retangulares cortadas de maneira não-guilhotinada, a partir de uma placa retangular; Cherri and Vianna (2012) abordam o problema de corte bidimensional com itens irregulares, adaptando a abordagem em Grafo E/OU de Morabito, ademais consideram padrões de cortes guilhotinados e não guilhotinados;

Já em Muzulon and Verga (2019), os autores adotam a heurística *First Fit Decreasing Height* (FFDH), em que o estudo analisa diferentes estratégias de geração de padrões de corte, considerando cortes exclusivamente guilhotinados, respeitando a restrição de orientação das

peças; Bischoff and Carvalho (2012) utilizam a meta-heurística GRASP combinado com técnicas de encaixe, para a organização e corte bidimensional de peças retangulares; Assis (2019) busca resolver o problema de corte de estoque bidimensional por meio da geração de padrões de corte guilhotinados ortogonais em 2-estágios restritos, a partir da heurística *HPD*, adaptada do método de Gilmore e Gomory 1965;

Conforme Costa and Poldi (2016), os autores propõem uma abordagem a partir da análise do modelo de Gilmore e Gomory, além de utilizar o modelo de fluxo em arcos (*Arcflow*) de Macedo adaptado, propondo um método para recuperar os padrões de corte, aplicando-o em peças retangulares com cortes guilhotinados em 2 estágios; Anjos *et al.* (2012) desenvolvem um algoritmo para peças retangulares com intuito de agrupar, selecionar e quantificar sobras e peças em classes bidimensionais para reaproveitamento, considerando no processo a orientação das peças; Marcelino (2019) propõe uma abordagem para a resolução do problema de corte bidimensional com itens regulares e irregulares, combinando seu método com CPLEX.

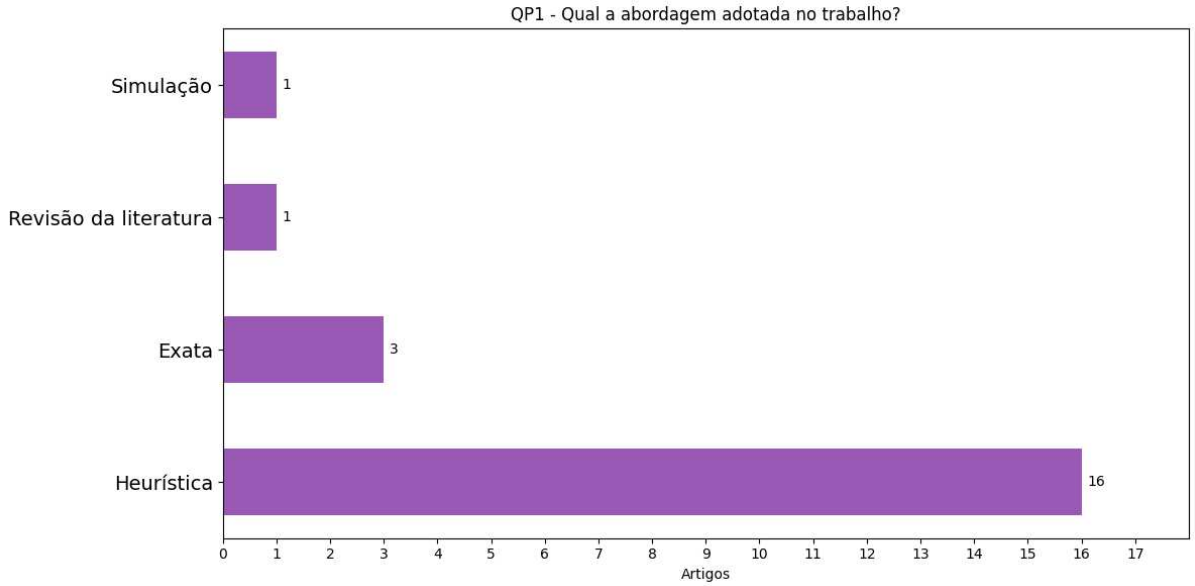
No trabalho de Martins (2010), o autor considera a orientação das peças e utiliza a heurística AFM-P, cujo objetivo é minimizar tanto o número de objetos cortados quanto o número de ciclos da serra; Oliveira (2018) apresenta uma forma nova de representação da solução e duas meta-heurísticas especializadas BRKGA e a Busca em Vizinhança com Variável Reduzida, para o PCE bidimensional não-guilhotinado. Mariano (2014) aborda o corte guilhotinado em peças retangulares, a partir da abordagem *Asynchronous Teams*, que utiliza agentes cooperativos baseado em uma heurística de caráter construtivo; Queiroz and Marques (2020) propõem o uso de um Algoritmo Genético Paralelo, implementado de forma distribuída com *Message Passing Interface* (MPI), avaliando operadores de seleção por roleta viciada sequencial e roleta viciada com e sem migração.

As abordagens que utilizaram algoritmos exatos correspondem a 13.69%, das quais são: Rocha (2015), que em seu trabalho adota o *Simplex* com geração de colunas, para o problema de corte guilhotinado e bidimensional; Nascimento (2022), em seu estudo, decompõe o problema em três etapas sucessivas e interdependentes, para resolver o corte bidimensional, mas que leva em consideração possíveis sobras aproveitáveis dos materiais utilizados, aplicando um *solver* exato, e posteriormente uma heurística *L- shaped* adaptada para tratar instâncias médias e grandes; Yanasse and Morabito (2013) consideram em seu trabalho padrões de corte guilhotinados de 2 estágios (exato e não exato), além de revisar e propor novos modelos lineares e não lineares inteiros, que são capazes de ser resolvidos por softwares de otimização, os quais

podem ser úteis para avaliar o desempenho de heurísticas.

Além disso, há apenas uma Revisão Sistemática da Literatura e Simulação, que são respectivamente os trabalhos de: Araújo, Araújo and Passos (2018), no qual realizam uma pesquisa com caráter qualitativo nas plataformas Capes, Scielo, Google Acadêmico e Scopus, em que dez artigos foram analisados com foco nos problemas de corte e suas aplicações; e Oliveira, Oliveira and Ghidini (2022), que utiliza o simulador FlexSim para criar um modelo de simulação, para avaliar se a solução encontrada pela otimização é viável, ademais um modelo de otimização linear inteiro é resolvido através de um método heurístico.

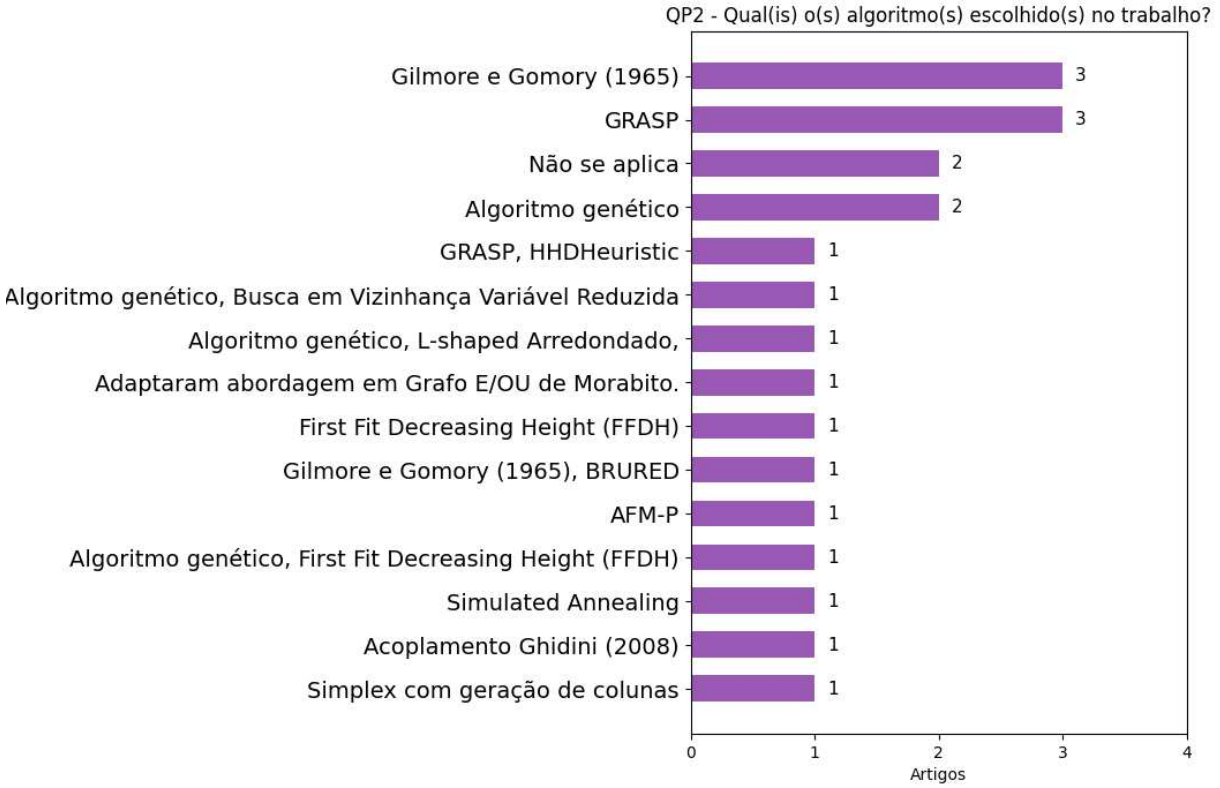
Figura 2: Resultados da questão de pesquisa 1 sobre a abordagem adotada no trabalho.



Fonte: Elaborada pela autora (2025).

Na Figura 3, é possível observar a variação que os trabalhos possuem quanto aos tipos de algoritmos usados. Neste sentido: Queiroz and Marques (2020), Nascimento (2022), Candido (2011), Queirós (2019) e Oliveira (2018) adotaram variações de Algoritmos Genéticos; Assis (2019), Costa and Poldi (2016), Yanasse and Morabito (2013) e Marcelino (2019) utilizaram abordagens adaptadas de Gilmore e Gomory; Rodrigues and Amaral (2016), Martinez (2014), Mariano (2014) e Bischoff and Carvalho (2012) abordaram a meta-heurística GRASP; Cherri and Vianna (2012) adapta a abordagem em Grafo E/OU de Morabito. Sepulveda (2013) utiliza o algoritmo *Simulated Anneling*; Martins (2010) emprega o algoritmo AFM-P; por fim, os dois últimos trabalhos marcados como não se aplica correspondem à simulação de Oliveira, Oliveira and Ghidini (2022) e a Revisão Sistemática da Literatura de Araújo, Araújo and Passos (2018).

Figura 3: Resultado da questão de pesquisa 2 sobre algoritmo(s) escolhido(s) no trabalho.

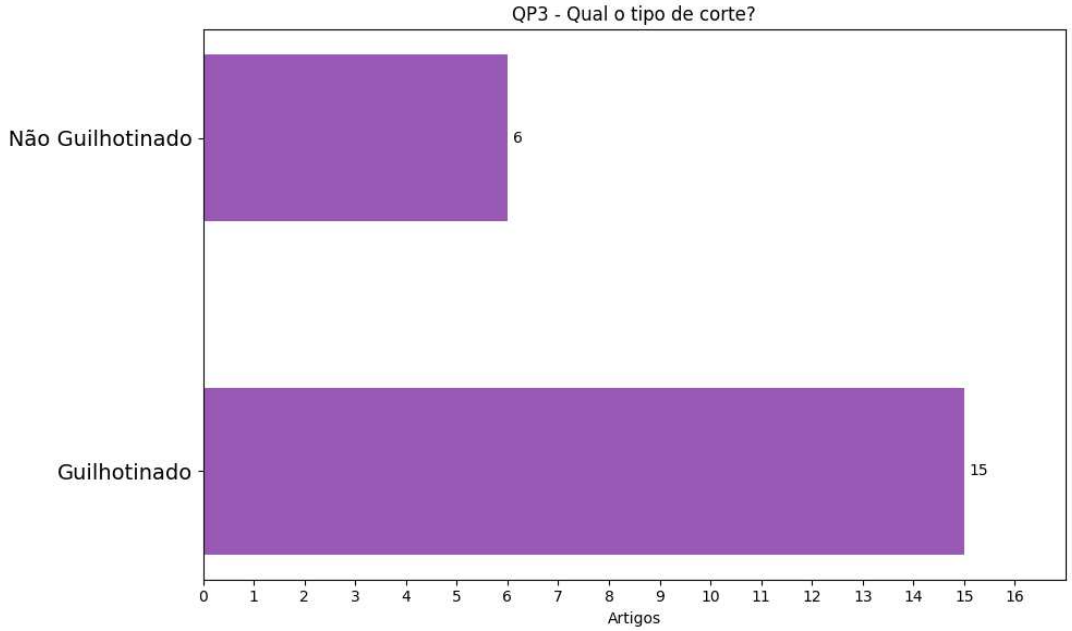


Fonte: Elaborada pela autora (2025).

Na Figura 4, é possível observar que 15 trabalhos tratam de cortes do tipo guilhotinado, os quais são abordados em: Oliveira, Oliveira and Ghidini (2022), Araújo, Araújo and Passos (2018), Candido (2011), Queirós (2019), Martinez (2014), Rocha (2015), Muzulon and Verga (2019), Bischoff and Carvalho (2012), Assis (2019), Costa and Poldi (2016), Anjos *et al.* (2012), Nascimento (2022), Mariano (2014), Yanasse and Morabito (2013) e Queiroz and Marques (2020). E os outros seis estudos são do tipo não-guilhotinado, os quais são mostrados nos trabalhos de: Sepulveda (2013), Rodrigues and Amaral (2016), Cherri and Vianna (2012), Marcelino (2019), Oliveira (2018) e Martins (2010).

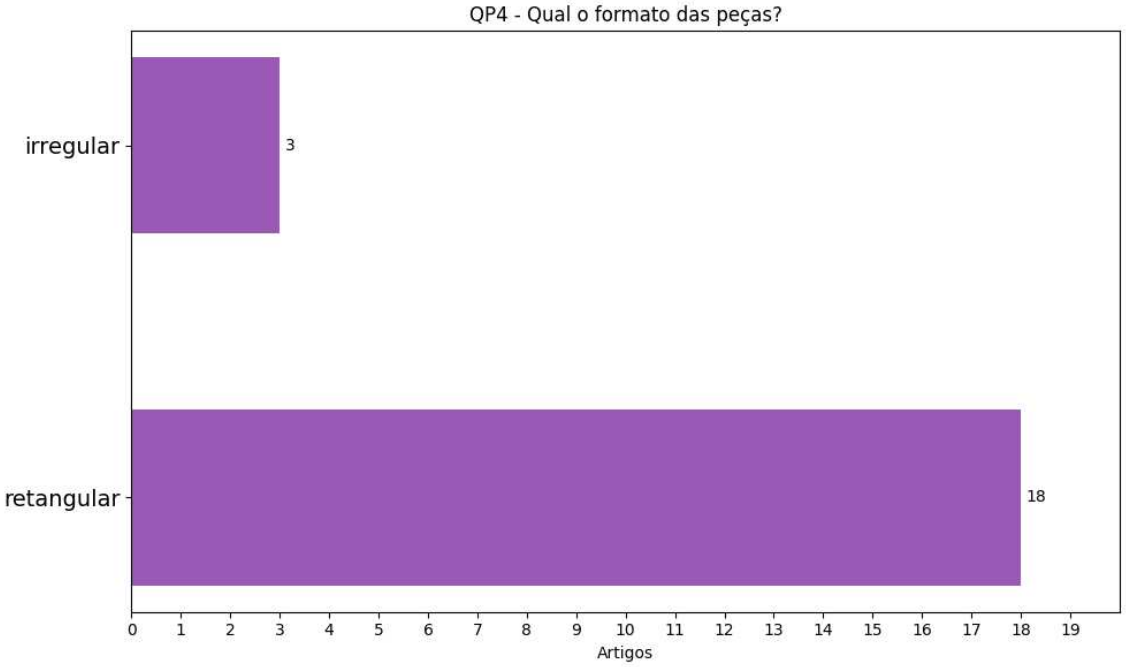
Na Figura 5, a maioria dos estudos retornados pela RSL são de peças cujo formato é retangular, com 18 artigos que se encaixam nessa categoria: Oliveira, Oliveira and Ghidini (2022), Sepulveda (2013), Candido (2011), Rodrigues and Amaral (2016), Queirós (2019), Martinez (2014), Rocha (2015), Muzulon and Verga (2019), Bischoff and Carvalho (2012), Costa and Poldi (2016), Assis (2019), Anjos *et al.* (2012), Martins (2010), Nascimento (2022), Oliveira (2018), Oliveira, Oliveira and Ghidini (2022), Yanasse and Morabito (2013) e Queiroz and Marques (2020). Os outros trabalhos consideram peças irregulares, como em: Marcelino (2019), Cherri and Vianna (2012) e Araújo, Araújo and Passos (2018).

Figura 4: Resultados da questão de pesquisa 3 sobre o tipo de corte considerado.



Fonte: Elaborada pela autora (2025).

Figura 5: Resultado da questão de pesquisa 4 sobre o formato das peças.

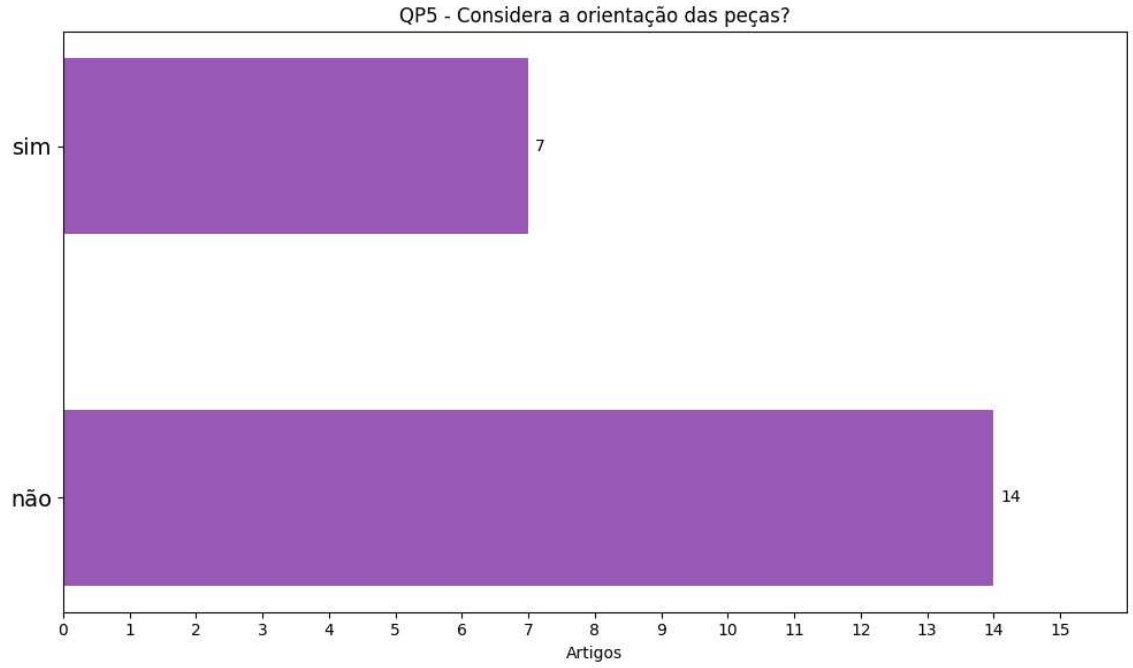


Fonte: Elaborada pela autora (2025).

Na Figura 6, verifica-se que 14 trabalhos não consideram a orientação das peças: Oliveira, Oliveira and Ghidini (2022), Araújo, Araújo and Passos (2018), Candido (2011), Rodrigues and Amaral (2016), Queirós (2019), Rocha (2015), Cherri and Vianna (2012), Bischoff and Carvalho (2012), Anjos *et al.* (2012), Assis (2019), Marcelino (2019), Nascimento (2022), Oliveira (2018), Mariano (2014) e Queiroz and Marques (2020). Em contrapartida, apenas sete

estudos exploram itens que são orientados, entre estes há: Candido (2011), Sepulveda (2013), Martinez (2014), Muzulon and Verga (2019), Costa and Poldi (2016), Martins (2010) e Yanasse and Morabito (2013).

Figura 6: Respostas da questão de pesquisa 5 sobre como os artigos consideram a orientação das peças.



Fonte: Elaborada pela autora (2025).

4 METODOLOGIA

Neste capítulo serão abordados os passos desenvolvidos durante a realização deste trabalho, incluindo: a revisão sistemática da literatura, as instâncias utilizadas, o algoritmo implementado e os resultados.

4.1 Passo 1 - Revisão da literatura

O primeiro passo executado para este trabalho foi a revisão sistemática da literatura, com o propósito de mapear os principais trabalhos da literatura atual que investigam os problemas de corte. O objetivo desta etapa consiste em identificar as abordagens algorítmicas utilizadas, bem como suas limitações, para fundamentar teoricamente e metodologicamente a escolha e a modelagem do AG, principalmente levando em consideração a definição das configurações dos operadores propostos neste estudo. A condução é melhor detalhada no Capítulo 3.

Conforme a Tabela 1, ao todo, 75 trabalhos foram retornados, dos quais 60 estavam disponíveis para *download*, mas apenas 21 trabalhos primários atendiam aos critérios de inclusão e de exclusão, estes que foram aceitos e lidos, para que suas informações fossem colhidas para responder às questões de pesquisa anteriormente estabelecidas, descritas na Subseção 3.2.4.

4.2 Passo 2 - Instâncias

Para a avaliação do algoritmo proposto, fez-se necessário o uso de um conjunto de instâncias que permitisse analisar seu desempenho em diferentes escalas de complexidade. Essas instâncias foram obtidas a partir de conjuntos de peças de móveis solicitados em marcenarias. O conjunto utilizado possui 6 tipos diferentes de instâncias, nas quais cada um possui 8 tipos de variações diferentes, que representam a quantidade de vezes que o conjunto base de peças é repetido para formar o conjunto final. Na Tabela 2, é possível observar a organização da instância.

A primeira coluna representa o todos os seis tipos de instâncias, as quais possuem respectivamente 10, 19, 19, 19, 39 e 106 de peças únicas, que se repetem 1, 5, 15, 20, 25, 35 e 40 vezes. O total de peças depende da quantidade de repetição das peças que são únicas.

Tabela 2: Conjunto de instâncias para o estudo.

Tipo da instância	Quantidade de peças únicas	Quantidade de peças por variação
Tipo 1	10	V01: 10, V05: 50, V15: 150, V20: 200, V25: 250, V30: 300, V35: 350, V40: 400
Tipo 2	19	V01: 19, V05: 95, V15: 285, V20: 380, V25: 475, V30: 570, V35: 665, V40: 760
Tipo 3	19	V01: 19, V05: 95, V15: 285, V20: 380, V25: 475, V30: 570, V35: 665, V40: 760
Tipo 4	19	V01: 19, V05: 95, V15: 285, V20: 380, V25: 475, V30: 570, V35: 665, V40: 760
Tipo 5	39	V01: 39, V05: 195, V15: 585, V20: 780, V25: 975, V30: 1170, V35: 1365, V40: 1560
Tipo 6	106	V01: 106, V05: 530, V15: 1590, V20: 2120, V25: 2650, V30: 3180, V35: 3710, V40: 4240

Fonte: Elaborada pela autora (2026).

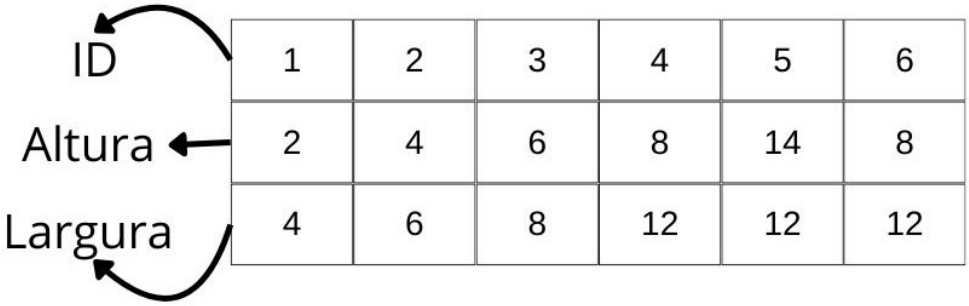
4.3 Passo 3 - Planejamento do algoritmo

Nesta seção será abordado o planejamento para a implementação da solução do problema de corte guilhotinado bidimensional em peças de madeira com orientação. O algoritmo escolhido foi o Algoritmo Genético, considerando as particularidades do problema, como a sua natureza combinatória, as restrições do corte e o objetivo de maximizar o saldo. O planejamento envolveu a definição da representação cromossomial, da função de avaliação, os operadores genéticos de seleção, mutação e *crossover*. Essas decisões foram tomadas a partir da análise da literatura, conforme detalhado no Capítulo 3, e também considerando as características do problema.

4.3.1 Representação cromossomial

A representação cromossomial é fundamental para o AG, dado que é a maneira pela qual os parâmetros do mundo real são traduzidos para a máquina (Linden, 2008). Neste trabalho, a representação cromossomial é baseada em um vetor de inteiros, no qual cada posição do vetor é um gene e corresponde a uma peça a ser cortada na placa de madeira. Além disso, cada peça contém um ID único, para garantir que todas as peças serão tratadas sem duplicidade ou exclusão, com suas respectivas alturas e larguras. Na Figura 7, demonstra-se um exemplo de representação cromossomial.

Figura 7: Representação cromossomial do algoritmo genético.

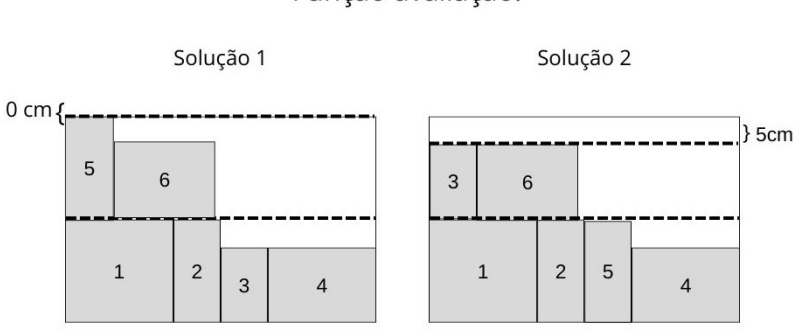


Fonte: Elaborada pela autora (2026).

4.3.2 Função de avaliação

A função de avaliação tem como objetivo mensurar a qualidade das soluções geradas pelo algoritmo (Pappa, 2002). Neste trabalho, ela é gerada a partir da quantidade de aproveitamento de madeira, ou seja, considera-se o quanto de material permanece disponível após o particionamento das peças. Para cada solução candidata, é calculada a sobra da placa, isto é, a área ou dimensão remanescente que não foi utilizada no processo de corte, para então mensurar a qualidade do resultado. Na Figura 8, verifica-se que a solução 1 possui avaliação zero, dado a distância da peça até a borda superior, já a solução 2 é melhor, pois todas as peças foram encaixadas e ainda obteve avaliação melhor, dado a distância de 5 centímetros entre a peça e a borda superior.

Figura 8: Exemplo de aplicação da função de avaliação.



Fonte: Elaborada pela autora (2026).

4.3.3 Operador de seleção

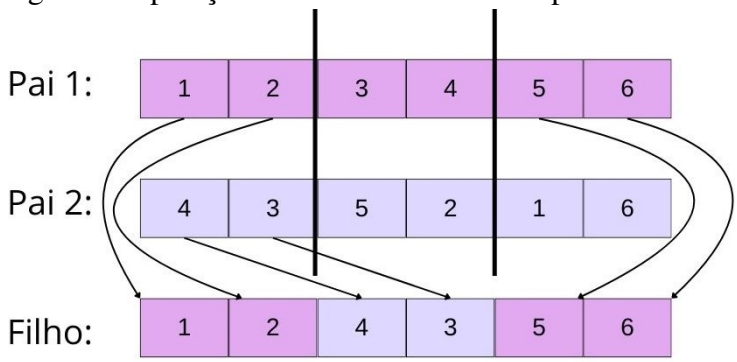
O operador de seleção tem como intuito simular a seleção natural que ocorre na natureza, favorecendo soluções com melhor desempenho, ao mesmo tempo em que mantém diversidade suficiente para explorar o espaço de busca (Cordeiro, 2007). Neste estudo, foram consideradas duas técnicas: a de seleção por roleta e a de seleção por torneio. Na seleção por roleta, cada indivíduo possui uma probabilidade proporcional ao seu valor de avaliação, deste modo, soluções que possuem uma avaliação maior possuem mais chances de serem selecionadas. Já o operador de seleção por torneio, um grupo de indivíduos é selecionado aleatoriamente da população, e aquele que possui maior avaliação é escolhido para reprodução.

4.3.4 Operador de crossover

O operador de *crossover* ou cruzamento, é responsável pelo processo de combinar características das soluções geradas, a partir da troca de genes dos indivíduos selecionados, denominados Pai 1 e Pai 2, assim produzindo filhos que herdam características de ambas as soluções (Reina, 2012). Este operador é baseado na reprodução de indivíduos biológicos, na qual há a recombinação das cadeias de DNA, dessa forma, os pais geram filhos a partir de sua combinação genética (Queiroz *et al.*, 2018).

Neste trabalho, serão adotados três tipos de operadores de cruzamento: o primeiro é o *crossover* com dois pontos, aplicado diretamente sobre o vetor de inteiros que representa a solução. Primeiramente, dois pontos de corte são definidos conforme o tamanho do cromossomo de posse dos dois pais escolhidos pelo operador de seleção e do ponto de corte. Em seguida, um filho será gerado combinando as informações genéticas dos pais selecionados. O processo de criação do novo filho será da seguinte maneira: a primeira parte, do início do cromossomo até o primeiro ponto de corte, herdará a informação genética do Pai 1. Em seguida, a parte intermediária, entre o primeiro e segundo corte, serão verificados quais IDs estão na região do Pai 1, para então ser verificada a ordem que eles aparecem no Pai 2, assim o Filho receberá os valores presentes na região entre os dois pontos de corte do Pai 1, mas com a sequência encontrada no Pai 2, por fim a parte final é preenchida com material genético do Pai 1. Na Figura 9 é mostrado um exemplo do funcionamento da aplicação do *crossover* dois pontos.

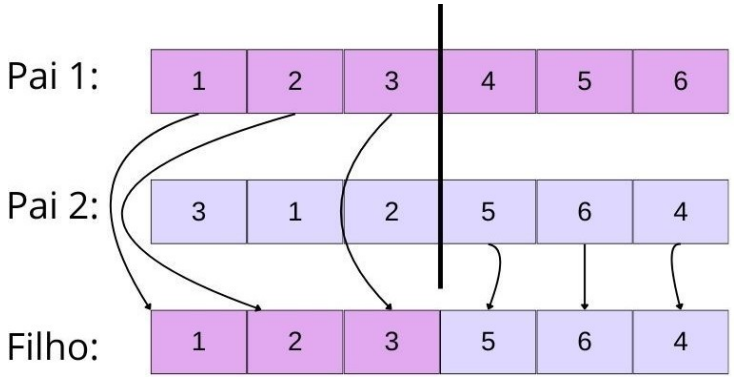
Figura 9: Operação de cruzamento de dois pontos.



Fonte: Elaborada pela autora (2026).

O segundo tipo de operador de cruzamento será o *crossover* de um ponto. Conforme apresentado na Figura 10, há dois indivíduos genitores, Pai 1 e Pai 2, os quais ambos sofrem corte na mesma posição, os dividindo em partes iguais, assim o Filho receberá em sua primeira metade o material genético do primeiro Pai, e sua segunda metade será formada pelos IDs das peças restantes conforme a ordem encontrada no Pai 2.

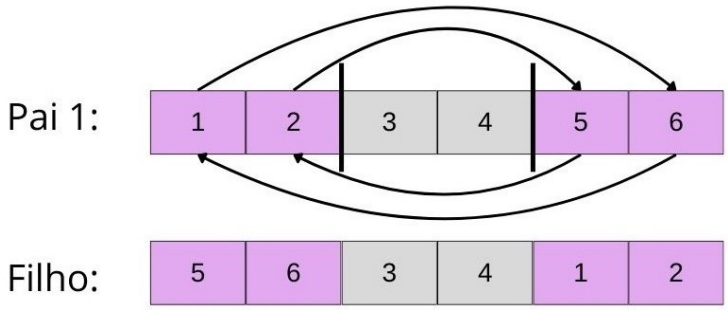
Figura 10: Operação de cruzamento de um ponto.



Fonte: Elaborada pela autora (2026).

Por fim, há o terceiro tipo de operador de cruzamento, o qual atua apenas sobre o Pai 1, que é seccionado em dois pontos, dividindo o cromossomo em três partes. Na Figura 11 pode-se observar a geração do indivíduo Filho: sua parte intermediária recebe a informação genética entre os dois cortes realizados no Pai, porém a sua primeira extremidade herda a parte final do cromossomo Pai, isto é, do inicio do segundo corte até o final do cromossomo, de maneira inversa, a parte final do cromossomo filho corresponde à parte inicial do indivíduo Pai, ou seja, do início do cromossomo até o primeiro corte.

Figura 11: Operação de cruzamento de dois pontos em pai único.

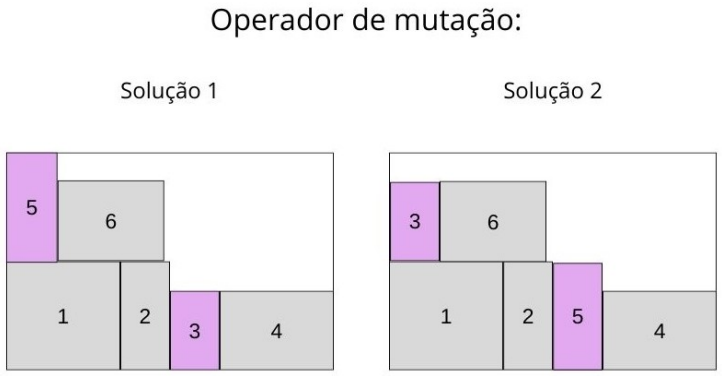


Fonte: Elaborada pela autora (2026).

4.3.5 Operador de mutação

O operador de mutação tem como objetivo aumentar a variabilidade dos cromossomos presentes na população, garantindo a diversidade genética da solução (Rocha, 2023). Na Figura 12 é possível observar o operador de mutação abordado, no qual há a troca de posição de peças, sem rotacioná-las, dada a restrição das peças serem orientadas. O operador é aplicado com uma taxa fixa de, por exemplo, 0.1%, com o intuito de que apenas uma parcela dos indivíduos sofra mutações a cada geração.

Figura 12: Operação de mutação em um cromossomo.



Fonte: Elaborada pela autora (2026).

A partir da combinação desses operadores espera-se abranger as melhores soluções.

4.4 Passo 4 - Análise comparativa dos algoritmos

Nesta etapa, serão realizadas as comparações dos algoritmos genéticos que serão compostos pelos operadores descritos anteriormente. No Quadro 2, é possível observar os operadores genéticos de cada um dos algoritmos que serão executados.

Quadro 2: Descrição dos operadores a serem utilizados neste trabalho

Experimento	**Operador de Seleção**	**Operador de *Crossover***	**Operador de Mutação**
Algoritmo 1	Método da Roleta	*Crossover* Dois Pontos	Uniforme
Algoritmo 2	Método da Roleta	*Crossover* Um Ponto	Uniforme
Algoritmo 3	Método da Roleta	*Crossover* Pai Único	Uniforme
Algoritmo 4	Método do Torneio	*Crossover* Dois Pontos	Uniforme
Algoritmo 5	Método do Torneio	*Crossover* Um Ponto	Uniforme
Algoritmo 6	Método do Torneio	*Crossover* Pai Único	Uniforme

Fonte: Elaborado pela autora (2026).

4.4.1 Passo 5 - Construção do Algoritmo

Para a construção do algoritmo genético, a solução foi organizada nas seguintes funções principais: leitura dos arquivos das instâncias; construção do *layout* da solução; função de aptidão; os métodos de seleção Roleta e Torneio; os operadores de Cruzamento Um Ponto, Dois Pontos e Pai Único; Mutação; e por fim o laço principal do algoritmo genético.

Primeiramente, o Algoritmo 2 executa a leitura das instâncias. Os arquivos possuem um formato padrão, em que: a primeira linha corresponde à quantidade de peças que a instância possui, a segunda linha contém as dimensões em largura e altura do painel de madeira, e as outras correspondem às respectivas largura e altura de cada peça. A função Ler Instância retorna as dimensões da peça e uma lista de objetos do tipo peça, na qual cada uma possui um identificador único, largura e altura. Esse identificador é essencial, pois o cromossomo do algoritmo genético será uma permutação desses identificadores, representando a ordem de inserção das peças na placa de madeira.

Visto que os dados já foram carregados, o algoritmo terá que transformar o cromossomo, ou seja, a combinação das peças, em *layout* físico que seja viável no painel de madeira, respeitando o corte guilhotinado. Dessa forma, a função Posicionar Peças foi desenvolvida, baseando-se na heurística *Shelf Packing* (Empacotamento em Prateleiras), que consiste em as peças serem inseridas uma a uma, na ordem definida pelo cromossomo, da direita para a esquerda e de baixo para cima. Nesse sentido, a prateleira é uma "linha" horizontal cuja altura é definida pela peça mais alta nela contida, atravessando um extremo a outro da placa. Quando uma peça não pode ser acomodada na largura restante da prateleira atual, esta não é mais considerada, e uma nova prateleira é iniciada imediatamente acima da anterior, levando em consideração a peça de maior altura, recomeçando o preenchimento a partir da borda esquerda. O processo de

consolidação da prateleira garante que todas as separações entre prateleiras sejam linhas retas que atravessam toda a largura da placa, satisfazendo assim a condição de corte guilhotinado.

Algoritmo 2: LERINSTANCIA

```
   Entrada: arquivo da instância
   Saída: larguraPlaca, alturaPlaca, lista de peças (Peca)
 1 início
 2 |  abrir arquivo em caminho;
 3 |  ler primeira linha → n;
 4 |  ler segunda linha → L, H;
 5 |  pecas ← lista vazia;
 6 |  para i ← 1 é n faça
 7 |  |  ler próxima linha → w, h;
 8 |  |  adicionar Peca(id=i, largura=w, altura=h) a pecas;
 9 |  fim
10 |  retorna L, H, pecas;
11 fim
```

Fonte: Elaborado pela autora (2026).

A função percorre o cromossomo e, para cada peça identificada, tenta: encaixá-la, se a peça couber horizontalmente no espaço da prateleira atual, ela é então posicionada levando em consideração as coordenadas (x, y), a variável x é deslocada para a direita, a largura da prateleira atual é decrescida e a altura é eventualmente atualizada para a peça mais alta. Porém, se uma peça não couber na largura restante, a prateleira será considerada fechada e então uma nova prateleira vazia é inicializada com coordenadas $x = 0$, a nova prateleira utilizará a coordenada y para saber em que posição a nova prateleira está na placa, e a largura da prateleira será redefinida para a largura do painel de madeira, então a mesma peça que causou o fechamento é testada, e o ciclo de encaixe de peças reinicia com o restante das peças.

Ao final do processamento de todos os genes, como é possível observar no Algoritmo 3, a função retorna a lista *pecasPosicionadas*, contendo as peças posicionadas com suas coordenadas (x, y) na placa. Essa lista servirá de base para o cálculo do *fitness*.

A qualidade das soluções geradas é medida pela função de Aptidão, presente no Algoritmo 4, definida como a altura restante na placa após o encaixe da última peça. O objetivo é minimizar a altura total, reduzindo desperdício e consequentemente fazendo que com essa sobra seja reutilizada para encaixar outras peças. Especificamente, calcula-se a máxima coordenada y de topo ($p.y + p.altura$) entre todas as peças posicionadas (linha 5) e subtrai-se esse valor da altura total da placa. Quanto maior for a sobra, melhor o indivíduo. Se a lista de peças posicionadas estiver vazia, o *fitness* é zero, representando a pior solução possível.

Algoritmo 3: POSICIONARPRATELEIRAS

Entrada: genes, pecas, larguraPlaca, alturaPlaca

Saída: lista de pecasposicionadas

```
1  início
2      para cada cada peça p contida na lista pecas faça
3          pecasPorId[p.id] ← p;
4      fim
5      pecasPosicionadas ← ∅; y ← 0; x ← 0; altPrat ← 0; largRest ← largPlaca;
6      para cada id ∈ genes faça
7          p ← pecasPorId[id];
8          se y + p.h > alturaPlaca então
9              continuar;
10         fim
11         se p.w > largRest então
12             y ← y + altPrat;
13             x ← 0;
14             largRest ← largPlaca;
15             altPrat ← 0;
16             se y + p.h > alturaPlaca ou p.w > largRest então
17                 continuar;
18             fim
19         fim
20         adicionar pecaPosicionada(id, p.w, p.h, x, y) a pecasPosicionadas;
21         x ← x + p.w;
22         largRest ← largRest − p.w;
23         altPrat ← max(altPrat, p.h);
24     fim
25     retorna pecasPosicionadas;
26 fim
```

Fonte: Elaborado pela autora (2026).

Algoritmo 4: CALCULARAPTIDAO

Entrada: pecascolocadas, totalPecas, larguraPlaca, alturaPlaca

Saída: altura restante

```
1 início
2     se pecasPosicionadas está vazia então
3         retorna 0;
4     fim
5     maxY ← máximo de p.y + p.altura para p em pecasPosicionadas ;
6     retorna alturaPlaca − maxY;
7 fim
```

Fonte: Elaborado pela autora (2026).

O próximo passo do algoritmo é criar a população inicial, para isso a função *GerarPopulaçaoInicial*, no Algoritmo 5, recebe a quantidade de indivíduos iniciais, os identificadores das peças e a semente. Cada indivíduo possui um cromossomo gerado aleatoriamente como uma permutação dos identificadores das peças. Ao final, a função devolve uma lista com os indivíduos gerados.

Algoritmo 5: GERARPOPULACAOINICIAL

Entrada: tamanhoPop, idsPecas, semente

Saída: lista de Individuos

```
1  início
2      se semente ≠ nulo então
3          definir semente ;
4      fim
5      pop ← lista vazia;
6      para i ← 1 é tamanhoPop faça
7          genes ← cópia de idsPecas embaralhada aleatoriamente;
8          adicionar novo Individuo(genes) a pop;
9      fim
10     retorna pop;
11 fim
```

Fonte: Elaborado pela autora (2026).

Dois métodos de seleção foram implementados: Roleta e Torneio. A seleção por Roleta, descrita no Algoritmo 6, associa cada indivíduo a uma probabilidade de seleção levando em consideração a Aptidão (*fitness*) que ele possui. Para tal, calcula-se o somatório total dos *fitness* da população, sorteia-se um ponto uniforme nesse intervalo e percorrem-se os indivíduos acumulando seus valores até que o ponto seja atingido. Dessa forma, favorecendo indivíduos que são mais fortes, isto é, possuem uma avaliação melhor.

Algoritmo 6: SELECAOROLETA

```
Entrada: populacaoAvaliada
Saída: individuoSelecionado
1  início
2      total ← 0;
3      para cada ind ∈ populacaoAvaliada faça
4          total ← total + ind.fitness;
5      fim
6      se total = 0 então
7          retorna indivíduo aleatório da populacaoAvaliada;
8      fim
9      ponto ← aleatorioUniforme(0, total);
10     acumulado ← 0;
11     para cada ind ∈ populacaoAvaliada faça
12         acumulado ← acumulado + ind.fitness;
13         se acumulado ≥ ponto então
14             retorna ind;
15         fim
16     fim
17     retorna último indivíduo da populacaoAvaliada;
18 fim
```

Fonte: Elaborado pela autora (2026).

Já a seleção por Torneio, conforme o Algoritmo 7, funciona a partir da escolha de k indivíduos sorteados aleatoriamente, com $k = 3$ para permitir que indivíduos com menor aptidão ainda tenham chances de serem escolhidos para competir. Ao final do torneio, o indivíduo com

melhor avaliação é escolhido.

Algoritmo 7: SELECAOTORNEIO

Entrada: população, k

Saída: indivíduo vencedor

```
1 início
2     candidatos ← amostra aleatória de k indivíduos da população;
3     retorna indivíduo de candidatos com o maior fitness;
4 fim
```

Fonte: Elaborado pela autora (2026).

Quanto aos operadores de cruzamento, para recombinar o material genético dos pais e gerar novos indivíduos, foram implementados três operadores de cruzamento. O primeiro corresponde ao operador Cruzamento Dois Pontos (Algoritmo 8), que atua da seguinte forma: primeiro ocorre o sorteio dos pontos de corte, no qual são gerados aleatoriamente dois índices inteiros a e b, com $0 \leq a < b \leq n$, em que n é o número total de genes (peças). O índice a é escolhido no intervalo $[0, n-2]$ e b no intervalo $[a+1, n]$. Esses dois pontos delimitam a região do corte e resultam em três regiões no cromossomo, a primeira é a região esquerda (posições 0 a $a-1$), a segunda é a região central (posições a e $b-1$) e a terceira é a região direita (posições b a $n-1$).

Posteriormente ocorre a herança das extremidades do primeiro pai: as regiões extremas, esquerda e direita do filho, recebem diretamente os genes do primeiro pai (*pai1*) exatamente na mesma ordem. Com isso, a parte externa do cromossomo do filho é idêntica à do *pai1*. Além disso, também ocorre a identificação do conjunto de genes da região central do *pai1*, em que se é coletado o conjunto dos IDs que ocupam as posições a até $b-1$ em *pai1*. Esse conjunto representa exatamente as peças que deverão aparecer na região central do filho, porém rearranjadas segundo a ordem do segundo pai.

O próximo passo é a obtenção da ordem do segundo pai para esse conjunto. Para isso, percorre-se *pai2* do início ao fim e selecionam-se, na ordem em que aparecem, todos os genes que pertencem à parte central do filho. Esses genes formam uma lista ordenada L, de mesmo tamanho que a região central. Logo após determinar a ordem de preenchimento da região central do filho, realiza-se a atribuição sequencial do gene correspondente, para cada posição i variando de a até $b-1$, atribui-se ao filho o gene correspondente de L.

Algoritmo 8: CRUZAMENTODOISPONTOS

Entrada: pai1, pai2

Saída: filho

início
  $n \leftarrow$ tamanho(pai1);
  $a \leftarrow$ aleatórioInteiro(0, $n-2$);
  $b \leftarrow$ aleatórioInteiro($a+1$, n);
  $filho \leftarrow$ lista de tamanho n preenchida com -1;
  $filho[0..a-1] \leftarrow pai1[0..a-1]$;
  $filho[b..n-1] \leftarrow pai1[b..n-1]$;
  $genes\_meio \leftarrow$ conjunto de $pai1[a..b-1]$;
  $ordem\_p2 \leftarrow$ elementos de $pai2$ que estão em $genes\_meio$, na ordem original;
  para $i \leftarrow a$ *é* $b-1$ **faça**
    $filho[i] \leftarrow ordem\_p2[i-a]$;
  fim
  retorna *filho*;
fim

Fonte: Elaborado pela autora (2026).

O segundo operador é o Cruzamento Um Ponto (Algoritmo 9), este possui uma implementação mais simples, pois apenas divide o cromossomo em um único ponto e completa a informação restante com as informações genéticas do outro pai.

Inicialmente ocorre o sorteio do ponto de corte, em que escolhe-se aleatoriamente um índice *ponto* de corte no *pai1* entre 1 e $n-1$. Então a sequência do *pai1* é copiada para o filho, do início até $ponto-1$. Logo após, efetua-se o preenchimento da segunda parte do filho com a ordem do *pai2*, a partir do *ponto* de corte, o filho é completado percorrendo-se os genes do *pai2* e inserindo, na ordem em que aparecem, aqueles que ainda não foram copiados, ou seja, que não pertencem ao prefixo herdado do pai1. Isso assegura que todos os IDs aparecem exatamente uma vez.

Algoritmo 9: CRUZAMENTOUMPONTO

Entrada: pai1, pai2

Saída: filho

```
1  início
2      n ← tamanho(pai1);
3      ponto ← aleatórioInteiro(1, n − 1);
4      filho ← lista de tamanho n preenchida com −1;
5      filho[0..ponto − 1] ← pai1[0..ponto − 1];
6      genes_copiados ← conjunto(filho[0..ponto − 1]);
7      idx ← ponto;
8      para cada gene ∈ pai2 faça
9          se gene ∉ genes_copiados então
10             filho[idx] ← gene;
11             idx ← idx + 1;
12         fim
13     fim
14     retorna filho;
15 fim
```

Fonte: Elaborado pela autora (2026).

O terceiro e último operador é o Cruzamento Pai Único (Algoritmo 10). Este utiliza apenas um pai para formar o filho, em que primeiramente são sorteados dois pontos de corte, a e b definidos de maneira aleatória, sendo $0 \leq a < b \leq n$. Dessa forma, o pai é dividido em três partes, a parte esquerda (*pai[0:a]*), central (*pai[a:b]*) e direita (*pai[b:n]*). Por fim, acontece a recombinação invertendo os extremos: o filho recebe em seu início a parte direita do pai (*pai[b:n]*), o trecho central continua seguindo a ordem do pai (*pai[a:b]*) e o intervalo final é preenchido com a parte direita do pai (*pai[0:a]*).

Algoritmo 10: CRUZAMENTOPAIUNICO

Entrada: pai

Saída: filho

início

$n \leftarrow$ tamanho(pai);

$a \leftarrow$ aleatórioInteiro(0, $n-2$);

$b \leftarrow$ aleatórioInteiro($a+1$, n);

$parte\_inicio\_filho \leftarrow$ pai_parte_fim[$b..n-1$];

$parte\_meio\_filho \leftarrow$ pai_parte_meio[$a..b-1$];

$parte_fim_filho \leftarrow$pai_parte_inicio[$0..a-1$];

retorna *filho*;

fim

Fonte: Elaborado pela autora(2026).

Por fim, o último operador empregado foi o de Mutação, com o objetivo de gerar pequenas perturbações aleatórias nos cromossomos, contribuindo para a manutenção da diversidade genética. Neste trabalho, emprega-se a mutação por troca (*swap*), apresentada no Algoritmo 11.

O funcionamento é simples: para cada indivíduo da população, gera-se um número aleatório uniforme entre 0 e 1. Se esse valor for inferior à taxa de mutação predefinida ($taxaMutacao$), duas posições distintas do cromossomo são sorteadas e os genes ali contidos são permutados. A taxa de mutação foi fixada em 0.1%. Como a permutação de dois elementos não altera o conjunto de IDs, a integridade da permutação é preservada nenhuma peça é removida ou duplicada.

Algoritmo 11: MUTACAOSWAP

Entrada: indivíduo, taxaMutacao

Saída: indivíduo

```
1 início
2     se aleatório() < taxaMutacao então
3         n ← tamanho(indivíduo.genes);
4         i, j ← duas posições aleatórias distintas em [0, n − 1];
5         trocar indivíduo.genes[i] e indivíduo.genes[j];
6     fim
7 fim
```

Fonte: Elaborado pela autora (2026).

A função principal do AG está representada no Algoritmo 12, responsável por orquestrar todas as funções descritas anteriormente, executando o ciclo evolutivo completo. Seus principais passos são:

- Inicialização: a população inicial de tamanho *tamanhoPop* é gerada por meio da função *GerarPopulacaoInicial* e, em seguida, todos os indivíduos são avaliados pela função *CalcularAptidão*. O melhor indivíduo dessa geração zero é armazenado como *melhorGlobal*.
- Laço geracional: o algoritmo executa *geracoes* (número de gerações definidas) ciclos evolutivos completos. Neste trabalho, a quantidade foi fixada em 100. Em cada geração, a população atual é ordenada por *aptidão* decrescente e uma nova população vazia é criada. Enquanto a nova população não atinge o tamanho *tamanhoPop*, dois pais são selecionados pelo método configurado (roleta ou torneio, conforme parâmetro *metodo_selecao*), então aplica-se o operador de cruzamento escolhido (*metodo_crossover*) sobre os genes dos pais, gerando um filho. Por fim, o filho resultante é submetido à mutação *MutacaoSwap* com taxa *taxaMutacao*, e é então inserido na nova população.
- Avaliação da nova geração: todos os indivíduos da nova população são avaliados.
- Atualização do melhor global: o indivíduo de maior *aptidão* da geração atual é comparado com *melhorGlobal* armazenado, se for superior, *melhorGlobal* é atualizado.
- Término da execução: após o término das gerações, o algoritmo retorna *melhorGlobal*, que contém a melhor solução encontrada, e também são coletadas estatísticas como tempo de execução, aproveitamento percentual e altura utilizada.

O algoritmo foi estruturado de forma a ser configurável, permitindo que a combinação

dos operadores atendessem os seis algoritmos definidos no Quadro 2.

Algoritmo 12: EXECUTARALGORITMOGENETICO

```
Entrada: idsPecas, pecas, larguraPlaca, alturaPlaca, tamanhoPop, geracoes,
         taxaMutacao, metodoSelecao, metodoCrossover, kTorneio, semente
Saída: melhor indivíduo encontrado
início
    populacao ← GerarPopulacaoInicial(tamanhoPop, idsPecas, semente);
    AvaliarPopulacao(populacao);
    melhorGlobal ← indivíduo de maior fitness da populacao;
    para g ← 1 é geracoes faça
        ordenar populacao por fitness decrescente;
        novaPopulacao ← lista vazia;
        enquanto tamanho(novaPopulacao) < tamanhoPop faça
            pai1 ← selecionar(populacao);
            pai2 ← selecionar(populacao);
            genesFilho ← cruzar(pai1.genes, pai2.genes);
            filho ← Individuo(genesFilho);
            MutacaoSwap(filho, taxaMutacao);
            adicionar filho a novaPopulacao;
        fim
        populacao ← novaPopulacao;
        AvaliarPopulacao(populacao);
        melhorGeracao ← indivíduo de maior fitness da populacao;
        se melhorGeracao.fitness > melhorGlobal.fitness então
            melhorGlobal ← cópia(melhorGeracao);
        fim
    fim
    retorna melhorGlobal;
fim
```

Fonte: Elaborado pela autora (2026).

5 RESULTADOS COMPUTACIONAIS

Neste capítulo são apresentados os resultados obtidos a partir da execução dos seis algoritmos propostos, descritos no Quadro 2, considerando as combinações entre os operadores de seleção e torneio. A análise foi feita com base em duas métricas principais: qualidade da solução encontrada, definida pela altura (sobra) da placa, e pelo tempo de execução que cada algoritmo obteve.

5.1 Configuração do Ambiente

Os algoritmos foram executados em um notebook Lenovo 82MF, sistema operacional Windows 11 Home, com processador AMD Ryzen 7 5700U, CPU 1.80 GHz X 8 e memória de 8 GB. Para a implementação do algoritmo foi utilizada a linguagem de programação Python versão 3.14.6. Além disso, o Python utiliza o algoritmo *Mersenne Twister* para geração de números pseudo-aleatórios. Para a geração das sementes foram utilizadas as casas decimais de π = 3.14159265358979323846, nas quais a cada três casas decimais, uma nova semente era gerada.

Os parâmetros do algoritmo genético foram configurados com:

- Tamanho da população: 100 indivíduos;
- Número de gerações: 100;
- Taxa de mutação: 0.1%;
- Tamanho do torneio (k): 3.

5.1.1 Execução do Algoritmo

Com o objetivo de analisar o desempenho dos algoritmos, estes receberam um conjunto de instâncias que representam peças reais, conforme descrito na Tabela 2. O nome de cada instância apresenta as informações sobre o tipo de instância e a quantidade de repetições daquele tipo de instância, por exemplo, a instância *m0_i25.txt*, possui um total de 10 peças únicas que se repetem 25 vezes. As instâncias foram definidas com o intuito de cobrir diferentes escalas de complexidade, contemplando os seis tipos de instâncias e as variações de repetição, totalizando 48 combinações distintas entre tipo e quantidade de repetições. O conjunto completo de instâncias utilizado é composto por: tipo 0 (m0), com 10 peças únicas; tipo 1 (m1), com 19 peças únicas; tipo 2 (m2), com 19 peças únicas; tipo 3 (m3), com 19 peças únicas; tipo 4 (m4), com 39 peças únicas; e o tipo 5 (m5), com 106 peças únicas. Que se repetem 1, 5, 15, 20, 25, 30,

35 e 40 vezes no total. Cada instância foi executada 30 vezes para cada um dos seis algoritmos, seguindo os parâmetros definidos anteriormente.

A Tabela 3 apresenta os tempos médios das 30 execuções obtidos, em segundos, e a sobra de altura para a combinação entre as instâncias e os algoritmos.

Tabela 3: Médias de tempo de execução (s) e sobra de altura por instância e algoritmo.

	Algoritmo 1		Algoritmo 2		Algoritmo 3		Algoritmo 4		Algoritmo 5		Algoritmo 6	
Instância	Tempo	Sobra	Tempo	Sobra	Tempo	Sobra	Tempo	Sobra	Tempo	Sobra	Tempo	Sobra
m0_i1	0.2321	274640	0.2308	274640	0.2291	274640	0.1234	274640	0.1160	274640	0.1136	274640
m0_i5	0.4505	824640	0.4465	824640	0.4322	824640	0.3427	824640	0.3322	824640	0.3143	824640
m0_i15	1.0032	2199640	1.0011	2199640	0.9343	2199640	0.9029	2199640	0.8852	2199640	0.8214	2199640
m0_i20	1.2795	3024640	1.2759	3024640	1.1879	3024640	1.1808	3024640	1.1594	3024640	1.0771	3024640
m0_i25	1.6258	3574640	1.5432	3574640	1.4343	3574640	1.4617	3574640	1.4337	3574640	1.3251	3574640
m0_i30	1.9539	4399640	1.8348	4399640	1.7083	4399640	1.7473	4399640	1.7155	4399640	1.5871	4399640
m0_i35	2.1721	4949280	2.1575	4949280	1.9989	4949280	2.0813	4949280	2.0433	4949280	1.8869	4949280
m0_i40	2.4677	5774280	2.4720	5774280	2.2851	5774280	2.3998	5774280	2.3537	5774280	2.1748	5774280
m1_i1	0.2866	274549	0.2804	274549	0.2783	274549	0.1749	274549	0.1663	274549	0.1617	274549
m1_i5	0.9411	824549	0.7006	824549	0.6673	824549	0.6007	824549	0.5908	824549	0.5515	824549
m1_i15	1.7717	2199098.2	1.7775	2199098.4	1.6595	2199098.9	1.6933	2199098.2	1.6609	2199098.3	1.5433	2199098.9
m1_i20	2.4150	3024098	2.4029	3024098	2.2305	3024098	2.3313	3024098	2.2918	3024098	2.1171	3024098
m1_i25	3.0356	3573688.4	3.0042	3573719.8	2.7804	3573773.9	2.9208	3573709.4	2.8762	3573759.3	2.6639	3573788.4
m1_i30	3.5951	4398647	3.5784	4398647	3.2945	4398647	3.5188	4398647	3.4485	4398647	3.1792	4398647
m1_i35	4.1192	4948647	4.1047	4948647	3.7689	4948647	4.0444	4948647	3.9809	4948647	3.6607	4948647
m1_i40	5.1060	5773196.0	4.6381	5773196.2	4.2698	5773196.4	4.6089	5773196.0	4.5339	5773196.2	4.1739	5773196.5
m2_i1	0.3049	274525	0.2796	274525	0.2761	274525	0.1747	274525	0.1662	274525	0.1611	274525
m2_i5	0.7109	824525	0.6977	824525	0.6655	824525	0.5999	824525	0.5896	824525	0.5489	824525
m2_i15	1.8728	2199077	1.7507	2199077	1.6419	2199077	1.6742	2199077	1.6480	2199077	1.5302	2199080.2
m2_i20	2.4502	3024052.7	2.3639	3024052.7	2.2080	3024061.7	2.3052	3024052.7	2.2638	3024051.8	2.0972	3024065.3
m2_i25	3.1385	3574050	3.1426	3574050	2.7482	3574050	2.9109	3574050	2.8591	3574050	2.6412	3574050
m2_i30	3.6149	4398599.3	3.5989	4398600.2	3.2447	4398602	3.4636	4398599.3	3.4014	4398601.1	3.1420	4398602
m2_i35	4.2579	4948575	4.0353	4948575	3.7158	4948576.8	3.9961	4948575	3.9360	4948575	3.6199	4948577.7
m2_i40	4.7170	5773127	4.5763	5773127	4.2098	5773141.6	4.5426	5773127	4.4854	5773127	4.1132	5773137.5
m3_i1	0.2901	274550	0.2798	274550	0.2761	274550	0.1753	274550	0.1666	274550	0.1623	274550
m3_i5	0.7813	824550	0.7053	824550	0.6669	824550	0.6014	824550	0.6031	824550	0.5519	824550
m3_i15	1.7930	2199264.6	1.7652	2199285.5	1.6586	2199345.2	1.6845	2199272.5	1.6545	2199339.9	1.5444	2199364.1
m3_i20	2.4663	3024100	2.3935	3024100	2.2389	3024100	2.3318	3024100	2.2936	3024100	2.1243	3024100
m3_i25	3.0765	3574100	3.0010	3574100	2.7960	3574100	2.9600	3574100	2.9010	3574100	2.6803	3574100
m3_i30	3.6645	4398744.3	3.5692	4398789.7	3.3009	4398800	3.5203	4398747	3.4647	4398786.3	3.1999	4398800
m3_i35	4.2089	4948650	4.1038	4948650	3.7790	4948650	4.0504	4948650	4.0046	4948650	3.6808	4948650
m3_i40	4.8664	5773650	4.6694	5773650	4.2864	5773650	4.5925	5773650	4.5686	5773650	4.1918	5773650
m4_i1	0.4096	274360	0.3895	274360	0.3768	274360	0.2850	274360	0.2777	274360	0.2618	274360
m4_i5	1.2677	824360	1.2546	824360	1.1747	824360	1.1523	824360	1.1445	824360	1.0620	824360
m4_i15	3.7326	2198120.8	3.6280	2198144.6	3.3467	2198195.7	3.5396	2198165.1	3.5056	2198163.6	3.2331	2198214.0
m4_i20	4.7813	3022555.1	4.7299	3022566.3	4.3509	3022571.7	4.6395	3022565.4	4.6121	3022569.9	4.2407	3022600.0
m4_i25	5.9002	3571941.7	5.8137	3571946.1	5.3463	3571947.7	5.7094	3571954.7	5.7162	3571955.1	5.2305	3571986.2
m4_i30	6.9726	4396403.7	6.8891	4396473.7	6.3454	4396536.5	6.7797	4396396.1	6.7672	4396505.0	6.2503	4396541.9
m4_i35	8.0808	4946186	7.9773	4946186	7.3069	4946193.7	7.8828	4946193.8	7.8643	4946196.8	7.2268	4946260.3
m4_i40	9.3127	5770562.9	9.1447	5770555.8	8.3756	5770590.1	9.0518	5770576.6	9.0297	5770585.2	8.2836	5770657.7
m5_i1	0.7737	274360	0.7640	274360	0.7201	274360	0.6582	274360	0.6520	274360	0.6071	274360
m5_i5	3.3554	823290.9	3.3109	823294.4	3.1048	823336.7	3.2029	823291.2	3.1746	823307.8	2.9511	823380.0
m5_i15	9.5725	2195768.6	9.3416	2195772.2	8.5734	2195787.8	9.2026	2195886.1	9.2573	2195875.3	8.4596	2195995.1
m5_i20	14.3873	3019490.6	13.5627	3019495.7	12.4786	3019522.4	13.3637	3019637.0	13.4133	3019628.3	12.3913	3019785.2
m5_i25	16.7081	3568215.5	16.6538	3568227.4	15.3645	3568255.8	16.5167	3568408.3	16.5527	3568370.9	15.4118	3568564.2
m5_i30	20.3321	4391956.2	19.8091	4391963.6	18.5169	4391965	19.7253	4392137.9	19.8606	4392118.9	18.2531	4392323.7
m5_i35	23.4178	4940179.3	22.9079	4940178.7	21.4344	4940184.1	22.8018	4940412.7	22.8592	4940392.0	21.0801	4940538.3
m5_i40	27.1538	5763891.8	27.1628	5763881.2	25.4781	5763918.4	27.0042	5764161.0	27.0370	5764107.1	25.0055	5764299.0

Fonte: Elaborada pela autora (2026).

Ao avaliar o comportamento do algoritmo nas instâncias iniciais do tipo m0 (de *m0_i1* a *m0_i40*), observa-se que todas as seis variações convergiram para valores idênticos de sobra de altura. Porém, ao analisar o tempo de execução entre os seis algoritmos, os Algoritmos 4, 5 e 6 demonstraram uma redução de aproximadamente 50% no tempo de execução em relação aos Algoritmos 1, 2 e 3. E entre os melhores tempos, o Algoritmo 6 possui a melhor média.

À medida que as instâncias aumentam em quantidade de peças, a complexidade do problema mostra que a disparidade entre os seis algoritmos torna-se mais evidente. Na instância *m1_i25*, nota-se que o Algoritmo 6 passa a se destacar tanto na qualidade da solução, obtendo a maior sobra de altura (3573788.4) com o menor tempo de execução (2.6639 s). Porém, ao verificar o Algoritmo 1, é possível observar que ele possui o pior desempenho tanto em relação à sobra (3573688.4) como em tempo de execução (3.0356 s).

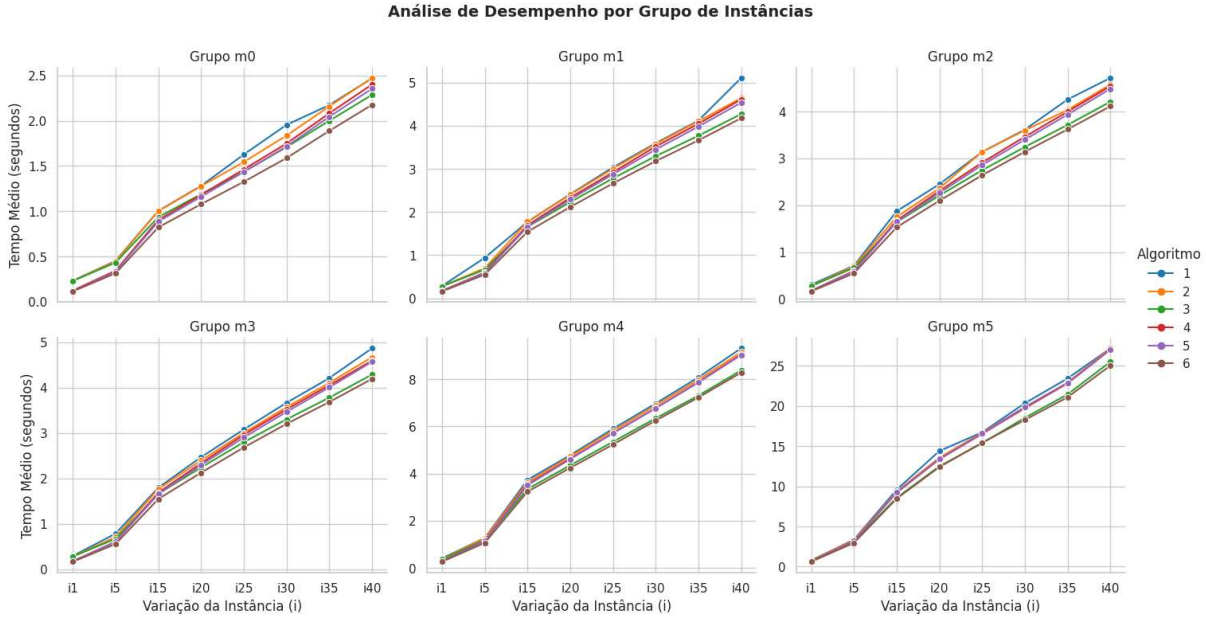
Ao se analisar as estatísticas de instâncias de tipo mais complexas, as da classe *m5*, torna-se ainda mais evidente que o Algoritmo 6 continua gerando os melhores resultados, em ambas as métricas. Na instância *m5_i40*, que possui a maior quantidade de peças de todas as instâncias (com 4240), o Algoritmo 6 executou em 25.0055 segundos e com a sobra de 5764299, enquanto o Algoritmo 4, que apresentou a segunda maior sobra (5764161) possuiu um tempo de execução de 27.0042, cerca de dois segundos a mais. Os Algoritmos 1 e 2 possuíram tempos similares de execução, com uma diferença de apenas 0.009 segundos de diferença, e com uma sobra de apenas 10.6, dessa forma os dois se consagram como algoritmos com as piores estatísticas levando em consideração a instância *m5_i40*.

Na Figura 13, evidencia-se a taxa de crescimento do tempo de execução à medida que o tamanho das instâncias expande-se de *m0* para *m5*. Ademais, fica mais evidente quais algoritmos apresentam um tempo de execução menor. Dessa forma, é possível constatar que os Algoritmos 3 e 6 inclinam-se de forma mais suave do que as demais. Esse comportamento gráfico demonstra que estas variações possuem uma melhor escalabilidade, tornando-as adequadas para cenários industriais reais, nos quais o volume de peças e chapas tende a ser elevado.

Dessa forma, o Algoritmo 6 consolida-se como o algoritmo com os melhores resultados globais, considerando tanto o tempo de execução quanto a sobra de altura. Ao analisar as instâncias em que houve diferença entre os algoritmos, observa-se que o Algoritmo 6 obteve a maior sobra em 19 das 48 instâncias em que os resultados não foram iguais (médias iguais) entre todas as configurações. O Algoritmo 3 obteve a maior sobra em apenas uma instância, enquanto os demais algoritmos não alcançaram o melhor resultado em nenhuma das instâncias analisadas.

Quanto à estabilidade dos algoritmos, esta pode ser observada por meio dos *Boxplots* das Figuras 14 e 15, considerando, respectivamente, a instância de menor e a de maior complexidade (*m0_i1* e *m5_i40*).

Figura 13: Gráfico de Linhas por Grupo de Instâncias.



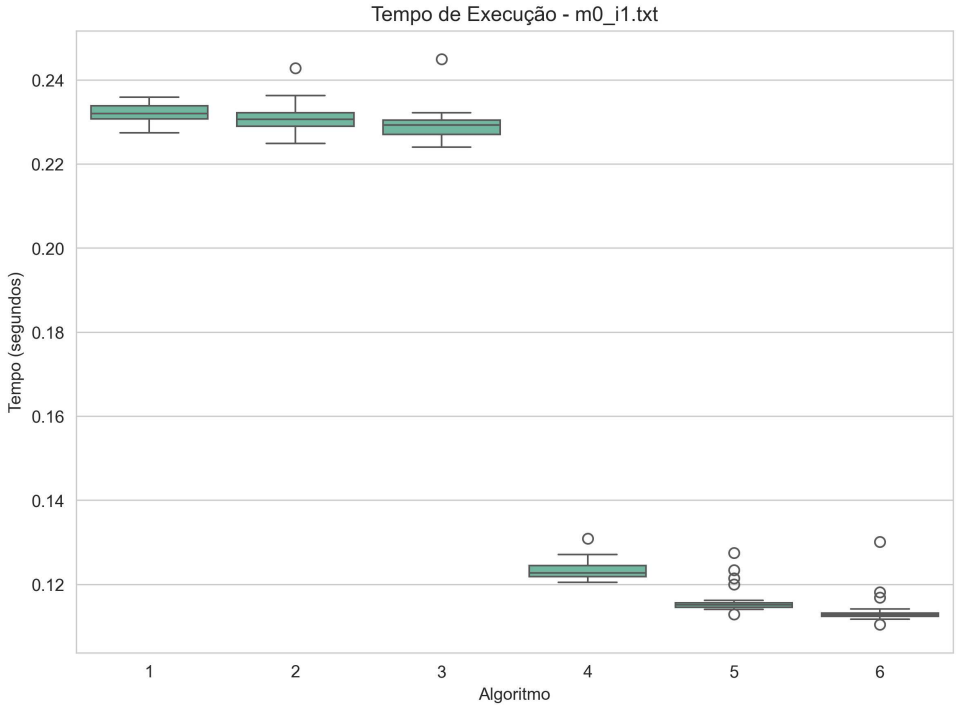
Fonte: Elaborada pela autora (2026).

A partir do *boxplot* referente à instância *m0_i1*, observa-se uma evidente divisão entre os Algoritmos 1, 2 e 3, e os Algoritmos 4, 5 e 6. O primeiro grupo apresenta as maiores medianas de tempo de execução, situando-se em torno de 0.23 segundos. Em contrapartida, o segundo grupo demonstra uma concentração das medianas na faixa de 0.12 segundos. Apesar da proximidade das medianas dentro de cada grupo, o Algoritmo 6 já se destaca por apresentar a menor mediana e o *Intervalo Interquartil* (IQR) mais reduzido, sugerindo menor variabilidade nos tempos de resposta para instâncias de pequeno porte, embora apresente alguns *outliers*.

Ao analisar a Figura 15, referente à instância de grande porte *m5_i40* (composta por 4240 peças), o cenário de desempenho muda parcialmente. Os Algoritmos 4 e 5, que eram competitivos na instância anterior, tiveram seus desempenhos piorados, passando a pertencer ao mesmo grupo dos Algoritmos 1 e 2, os quais apresentam medianas de aproximadamente 27.0 segundos. Nesse cenário, o Algoritmo 3 demonstra uma excelente escalabilidade, tornando-se o segundo melhor, com mediana de aproximadamente 25.5 segundos.

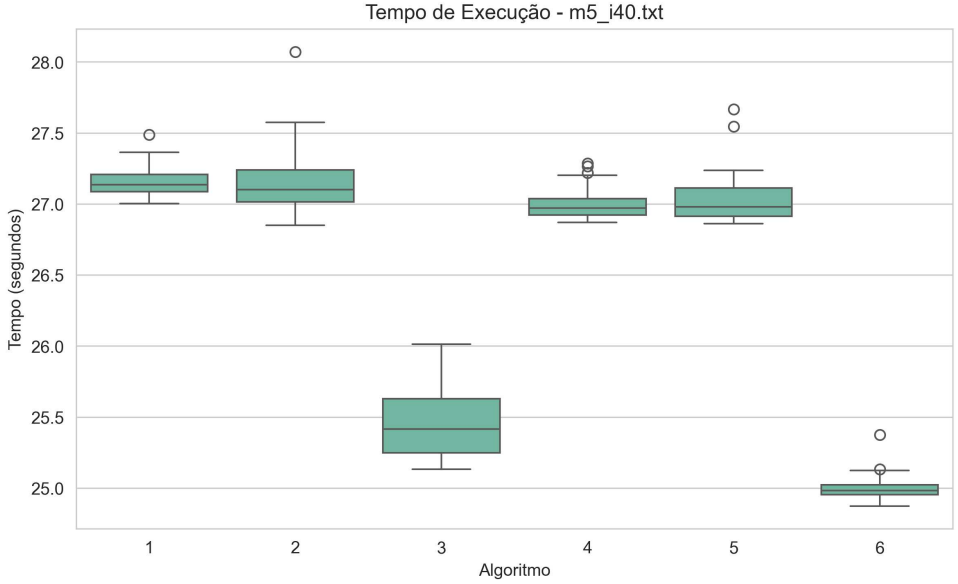
Nesse cenário, o Algoritmo 6 consolida-se como o mais eficiente também para problemas complexos, operando em cerca de 25.0 segundos. Ademais, seu IQR é o mais estreito entre todos os algoritmos, o que o caracteriza como o algoritmo mais estável e confiável, evidenciando que o tempo de execução apresenta baixa variação entre as 30 execuções realizadas.

Figura 14: *Boxplot* do tempo de cada algoritmo na instância m0_i1.



Fonte: Elaborada pela autora (2026).

Figura 15: *Boxplot* do tempo de cada algoritmo na instância m5_i40.



Fonte: Elaborada pela autora (2026).

5.1.2 Exemplo de uma Solução

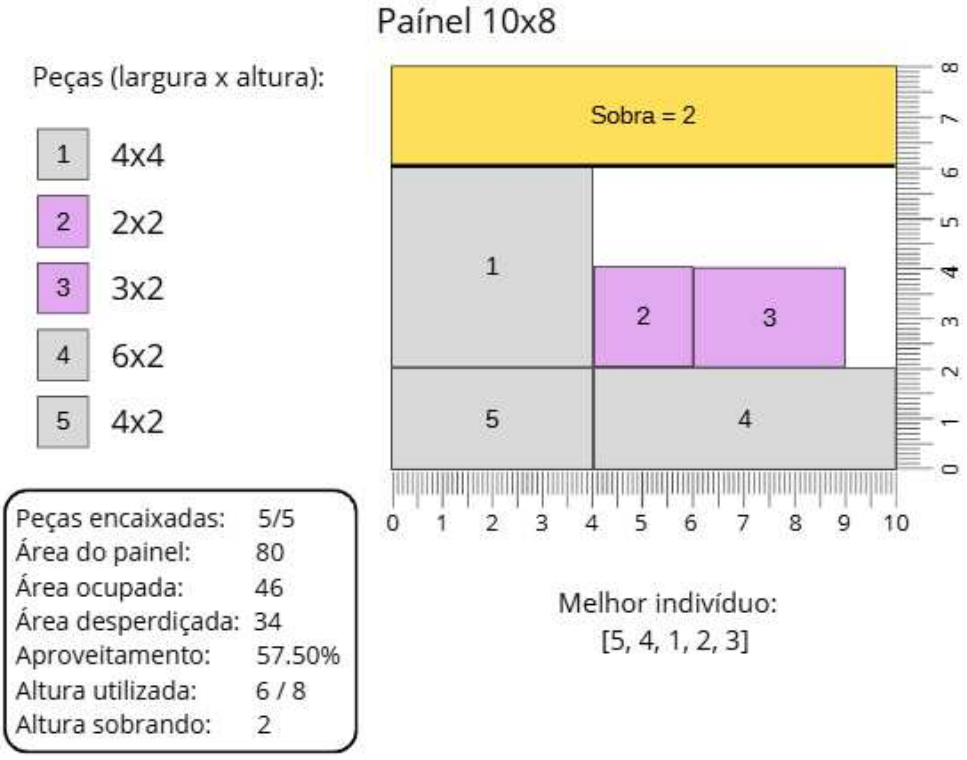
Para demonstrar visualmente o comportamento do algoritmo proposto e validar a implementação correta dos operadores de seleção, reprodução e mutação, esta seção apresenta um exemplo de solução para uma instância de pequeno porte, criada especificamente para a demonstração. A Figura 16 ilustra o resultado da execução do Algoritmo 6.

A instância utilizada para este exemplo foi definida com um painel de dimensões 10x8 e um total de 5 peças, conforme o arquivo de entrada abaixo:

```
5
10 8
4 4
2 2
3 2
6 2
4 2
```

A partir dos dados de entrada, o algoritmo realizou o posicionamento das peças no painel. O resultado gráfico obtido está representado na Figura 16, na qual verifica-se o *layout* final e a ausência de sobreposição entre as peças, obedecendo às restrições do problema de corte guilhotinado bidimensional.

Figura 16: Exemplo de solução gerada pelo AG.



Fonte: Elaborada pela autora (2026).

A partir da imagem, nota-se que a solução encaixou todas as peças dentro dos limites do painel, demonstrando a eficácia do método de posicionamento utilizado.

6 CONCLUSÃO

O presente trabalho teve como objetivo principal identificar a melhor configuração de um Algoritmo Genético aplicado ao problema de corte guilhotinado e bidimensional em placas de madeira, buscando maximizar o aproveitamento do painel e também minimizar o tempo de produção de soluções viáveis. Para isso, foram implementadas e comparadas seis variações do AG, combinando diferentes operadores de seleção e cruzamento, conforme os experimentos definidos no Quadro 2.

A partir da execução dos experimentos com as 48 instâncias, que possuíam níveis de complexidade distintos, os resultados demonstraram que a combinação de diferentes operadores genéticos influenciam significativamente o desempenho do algoritmo. A análise dos gráficos e da Tabela 3, demonstrou que o Algoritmo 6, composto pelo operador de seleção por Torneio e pelo operador de cruzamento de Pai Único, revelou-se como a combinação mais eficiente. Essa combinação demonstrou a melhor escalabilidade, visto que, apesar do incremento na complexidade das instâncias, o Algoritmo 6 continuou obtendo o menor tempo de execução, atingindo cerca de 25 segundos para a instância mais complexa, *m5_i40*, que continha 4240 peças, e a maior estabilidade, evidenciada pela amplitude IQR compactada. Além disso, ele demonstrou a melhor qualidade de soluções geradas, obtendo a maior "sobra de altura" (aproveitamento) em 19 das 48 instâncias analisadas.

Contudo, há uma limitação no modelo proposto. Corresponde à restrição de orientação das peças, já que elas não puderam sofrer rotações de 90 graus, o que, embora reflita a realidade de alguns móveis estampados, acaba por reduzir o espaço de busca do algoritmo e impede cenários mistos, em que algumas peças poderiam ser giradas. Consequentemente, o aproveitamento máximo da placa de madeira torna-se limitado. Além disso, o modelo foca em um ambiente estático, não acompanhando as oscilações em tempo real de uma linha de produção dinâmica que normalmente ocorre em fábrica de móveis.

Levando em consideração os resultados obtidos e a limitação encontrada, sugere-se como trabalho futuro: permitir que o algoritmo lide com peças orientadas e não orientadas; implementar uma interface gráfica com o objetivo de gerar a visualização 2D das soluções sugeridas pelo algoritmo; e comparar os algoritmos propostos neste trabalho com algoritmos consagrados na literatura.

Em resumo, este trabalho demonstrou que a implementação de um Algoritmo Genético com operadores sugeridos é uma abordagem viável, rápida e confiável para a resolução

do problema de corte guilhotinado. O Algoritmo 6 surge como a solução mais indicada para cenários industriais reais, onde o volume de peças e a necessidade de respostas rápidas são essenciais para a redução de desperdícios e de otimização da linha de produção.

REFERÊNCIAS

ANJOS, J. M. d. *et al.* **GERENCIAMENTO DAS SOBRAS DE PAINÉIS DE MADEIRA NA INDÚSTRIA MOVELEIRA.** 2012. Dissertação (Mestrado) — Pontifícia Universidade Católica de Goiás, 2012.

ARAÚJO, A. M.; ARAÚJO, D. de O.; PASSOS, F. G. dos. Análise das vantagens do uso da pesquisa operacional em problemas de corte: uma revisão sistemática da literatura. **Produção em Foco**, 2018.

ASSIS, N. S. **O problema de corte de estoque bidimensional: geração de padrões de corte 2-estágios restritos**. 2019. Dissertação (Mestrado) — Universidade Estadual Paulista (Unesp), 2019.

BISCHOFF, E.; WäSCHER, G. Cutting and packing. **European Journal of Operational Research**, 1995.

BISCHOFF, V.; CARVALHO, F. P. de. **A APLICAÇÃO DO ALGORITMO HEURÍSTICO BIDIMENSIONAL NO CORTE RACIONAL DE CHAPAS DE MADEIRA, NA CONSTRUÇÃO CIVIL**. 2012. Monografia (Bacharelado em Sistemas de Informação) — Faculdades Integradas de Taquara (FACCAT), Taquara - RS, 2012.

CANDIDO, L. C. X. Geração de padrões de corte em guilhotina bidimensional utilizando algoritmos genéticos e heurísticas de encaixe. **Programa de Pós-Graduação em Métodos Numéricos em Engenharia - Universidade Federal do Paraná**, 2011.

CHERRI, A.; VIANNA, A. Problemas de corte com itens irregulares. *In*: **XVI CLAIO/ XLIV SBPO - Congreso Latino-Iberoamericano de Investigación Operativa / Simpósio Brasileiro de Pesquisa Operacional**. [*S.l.: s.n.*], 2012.

CORDEIRO, A. dos S. **Otimização e melhoramento exergoeconômico de sistemas térmicos modelados em um simulador de processos utilizando métodos de busca direta e estocástico**. 2007. Tese (Doutorado) — UNIVERSIDADE FEDERAL DO RIO DE JANEIRO (UFRJ), 2007.

COSTA, L. L. S.; POLDI, K. **Um estudo sobre o problema de corte de estoque bidimensional 2-estágios**. 2016. Dissertação (Mestrado) — IMECC-UNICAMP, 2016.

FAO, F. **The global forest sector outlook 2050: Asseesing future demand and sources of timber for a sustainable economy**. [*S.l.*], 2022.

GILMORE, P. C.; GOMORY, R. E. A linear programming approach to the cutting-stock problem. **Operations Research (OR)**, 1961.

GRAMANI, M. C. G. **Problema de corte bidimensional guilhotinado e restrito em 2-estágios** 2018. 144 f. 1997. Dissertação (Mestrado) — Ciências da Computação e Matemática Computacional – Instituto de Ciências Matemáticas de São Carlos, 1997.

HOFFMANN, F. M. *et al.* Otimização de padrões de cortes bidimensionais guilhotinados restritos. **Revista ESPACIOS**, 2015.

KITCHENHAM, B.; CHARTERS, S. **Guidelines for performing Systematic Literature Reviews in Software Engineering**. [*S.l.*], 2007.

LINDEN, R. **Algoritmos genéticos (2a ediçao)**. [*S.l.: s.n.*]: Brasport, 2008.

MACEDO, M. A. *et al.* Identificação e classificação dos custos ambientais na indústria moveleira. *In*: **XXXIII Encontro Nacional de Engenharia de Produção**. [*S.l.: s.n.*], 2013.

MARCELINO, K. A. d. S. **Geração de padrões de corte bidimensionais com itens regulares e irregulares do tipo L**. 2019. Monografia (Bacharelado em Ciência da Computação) — Universidade Estadual Paulista (Unesp), Bauru - SP, 2019.

MARIANO, M. F. d. S. **Uma metaheurística a-teams aplicada ao problema do corte bidimensional guilhotinado**. 2014. Dissertação (Mestrado) — Universidade Federal Rural do Semi-Árido, Universidade do Estado do Rio Grande do Norte - Programa de Pós-Graduação em Ciência da Computação., 2014.

MARTINEZ, D. A. **Estudo dos problemas de corte e empacotamento**. 2014. Tese (Doutorado) — Universidade Estadual Paulista (Unesp), 2014.

MARTINS, A. T. **Estratégias para a redução de ciclos da serra no problema de corte de estoque na indústria moveleira**. 2010. Dissertação (Mestrado) — Universidade Estadual Paulista (Unesp), 2010.

MÓVEIS, B. **Dados do Setor 2024**. 2024. Relatório Setorial da Indústria de Móveis no Brasil. Available at: <hhttps://abimovel.com/capa/dados-do-setor/>. Access at: 20 out. 2025.

MUZULON, N.; VERGA, J. Otimizando recursos em uma indústria moveleira através do problema de corte bidimensional: uma abordagem utilizando first fit decreasing height (ffdh). *In*: **XXVI Simpósio de Engenharia de Produção**. [*S.l.: s.n.*], 2019.

NASCIMENTO, D. N. d. **O problema de corte bidimensional com sobras aproveitáveis e incerteza na demanda**. 2022. Tese (Doutorado) — Universidade Estadual Paulista (Unesp), 2022.

OLIVEIRA, E. V. d. **Otimização do problema de corte bidimensional não guilhotinado usando meta-heurísticas especializadas**. 2018. Tese (Doutorado) — Faculdade de Engenharia de Ilha Solteira – Universidade Estadual Paulista (Unesp), 2018.

OLIVEIRA, O. C. F. de; LOPES, I. E. C. Pesquisa operacional e a programação linear na minimização de custos e maximização de lucros: Um estudo de caso. **Revista Matemática e Educação**, 2022.

OLIVEIRA, P. M.; OLIVEIRA, W. A.; GHIDINI, C. T. L. S. SimulaÇÃo-otimizaÇÃo aplicada ao problema integrado na industria de mÓveis. **Pesquisa Operacional para o Desenvolvimento**, 2022.

PACHECO, M. A. C. *et al.* Algoritmos genéticos: princípios e aplicações. **ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro. Fonte desconhecida**, 1999.

PAPPA, G. L. **Seleção de atributos utilizando Algoritmos Genéticos multiobjetivos**. 2002. Dissertação (Mestrado) — Universidade Pontifícia Católica do Paraná–Programa de Pós-Graduação em Informática Aplicada, 2002.

PINTO, A. R. F.; MARTARELLI, N. J.; NAGANO, M. S. Algoritmo genético: principais gaps, trade-offs e perspectivas para futuras pesquisas. **Trends in Computational and Applied Mathematics**, 2022.

QUEIRÓS, I. P. C. **Problemas de corte de peças com incerteza na qualidade da matéria-prima-uma abordagem com algoritmos genéticos**. 2019. Dissertação (Mestrado) — Faculdade de Engenharia da Universidade do Porto (FEUP), 2019.

QUEIROZ, M. D. d. *et al.* **SmartSubway: um sistema colaborativo para apoiar o estudo da eficiência energética em trens urbanos no contexto de cidades inteligentes**. 2018. Dissertação (Mestrado) — Universidade Federal da Paraíba, 2018.

QUEIROZ, M. de; MARQUES, E. Identificação do operador de seleção do algoritmo genético paralelo para o problema de corte bidimensional. *In*: **Escola Regional de Computação Bahia, Alagoas e Sergipe (ERBASE)**. [*S.l.: s.n.*], 2020.

QUEIROZ, M. de *et al.* Um framework para apoiar especialistas no estudo da eficiência energética em trens urbanos. *In*: **Anais do XV Simpósio Brasileiro de Sistemas de Informação**. Porto Alegre, RS, Brasil: SBC, 2019. p. 32–39. ISSN 3086-4836. Available at: <https://sol.sbc.org.br/index.php/sbsi/article/view/13886>.

REINA, C. D. **Roteirização de veículos com janelas de tempo utilizando algoritmo genético.** 2012. Tese (Doutorado) — Universidade de São Paulo, 2012.

ROCHA, G. R. **Aplicação de algoritmo genético para roteirização e carregamento de veículo**. 2023. Monografia (Bacharelado em Ciência da Computação) — Universidade Estadual Paulista (Unesp), Bauru - SP, 2023.

ROCHA, R. F. **O Problema de Corte de Estoque numa Indústria Moveleira**. 2015. Dissertação (Mestrado) — Universidade Estadual Paulista "Julio de Mesquita Filho" - Instituto de Biociencias, Letras e Ciências Exatas - Campus de Sao José do Rio Preto, 2015.

RODRIGUES, T. N.; AMARAL, A. R. S. Meta-heur istica grasp aplicada ao problema do corte bidimensional nao-guilhotinado. *In*: **XLVIII SBPO - Simpósio Brasileiro de Pesquisa Operacional**. [*S.l.: s.n.*], 2016.

SEPULVEDA, G. P. L. **Solução do problema de corte bidimensional de peças retângulares tipo não-guilhotinado usando simulated annealing**. 2013. Dissertação (Mestrado) — Universidade Estadual Paulista (Unesp), 2013.

SILVA, E. M. da C. **Modelos e Métodos De Otimização Para Problemas De Corte e Empacotamento a Duas Dimensões**. 2011. Tese (Doutorado) — Universidade do Minho (Portugal), 2011.

SILVA, M. H. N. d. *et al.* **Aplicações de meta-heurísticas na resolução de problemas de sequenciamento de tarefas**. 2024. Monografia (Bacharelado em Engenharia de Transportes e Logística) — Universidade Federal de Santa Catarina - UFSC, Joinville - SC, 2024.

SILVEIRA, R. J.; MORABITO, R. Um método heurístico baseado em programação dinâmica para o problema de corte bidimensional guilhotinado restrito. **Gestão & Produção**, 2002.

VIANA, G. V. R. *et al.* Problemas de corte e empacotamento bidimensionais: um aplicativo inteligente para paletização de produtos. **Brazilian Journal of Development**, 2021.

VIANNA, A. C. G. **Problemas de corte e empacotamento: uma abordagem em grafo E/OU**. 2000. Tese (Doutorado) — Universidade de São Paulo, 2000.

WÄSCHER, G.; HAUSSNER, H.; SCHUMANN, H. An improved typology of cutting and packing problems. **European journal of operational research**, 2007.

YANASSE, H. H.; MORABITO, R. Modelos lineares e não lineares inteiros para problemas da mochila bidimensional restrita a 2 estágios. **Production**, 2013.