



UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO



VINICIUS MUNIZ MAGALHÃES

**UTILIZAÇÃO DE MÉTODOS META-HEURÍSTICOS PARA A RESOLUÇÃO DO
PROBLEMA DE MÁQUINAS PARALELAS COM RESTRIÇÃO DE RECURSOS**

OURO PRETO - MG

2026

Vinicius Muniz Magalhães

**UTILIZAÇÃO DE MÉTODOS META-HEURÍSTICOS PARA A RESOLUÇÃO DO
PROBLEMA DE MÁQUINAS PARALELAS COM RESTRIÇÃO DE RECURSOS**

Monografia apresentada ao Curso de Graduação
em Engenharia de Produção da Universidade
Federal de Ouro Preto como requisito parcial para
a obtenção do título de Engenheiro de Produção.

Orientador: Dr. Aloisio de Castro Gomes Junior

OURO PRETO – MG

2026

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

M188u Magalhaes, Vinicius Muniz.

Utilização de métodos meta-heurísticos para a resolução do problema de máquinas paralelas com restrição de recursos. [manuscrito] / Vinicius Muniz Magalhaes. - 2026.

93 f.: il.: color., gráf., tab.. + Quadro.

Orientador: Prof. Dr. Aloísio de Castro Gomes Junior.

Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Produção .

1. Planejamento e Controle da Produção (PCP). 2. Otimização combinatória. 3. Administração da produção. I. Gomes Junior, Aloísio de Castro. II. Universidade Federal de Ouro Preto. III. Título.

CDU 658.5

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716



FOLHA DE APROVAÇÃO

Vinicius Muniz Magalhães

Utilização de métodos meta-heurísticos para a resolução do problema de máquinas paralelas com restrição de recursos

Monografia apresentada ao **Curso de Engenharia de Produção** da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de **Bacharel em Engenharia de Produção**.

Aprovada em 01 de setembro de 2026

Membros da banca

Dr. Aloísio de Castro Gomes Júnior - Orientador(a) (Universidade Federal de Ouro Preto)
Dr. Helton Cristiano Oliveira e Gomes - (Universidade Federal de Ouro Preto)
Dra. Clarisse da Silva Vieira Camelo de Souza - (Universidade Federal de Ouro Preto)
Eng. Jéssica Natália Miranda Paiva - (Universidade Federal de Ouro Preto)

Dr. Aloísio de Castro Gomes Júnior, orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 01/09/2026.



Documento assinado eletronicamente por **Aloisio de Castro Gomes Junior, PROFESSOR DE MAGISTERIO SUPERIOR**, em 04/09/2026, às 15:34, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1185023** e o código CRC **76D3FBF3**.

À minha família, pelo amor, incentivo e confiança.

AGRADECIMENTOS

Primeiramente, agradeço a Deus, por me conceder saúde, força, sabedoria e por guiar meus passos ao longo desta jornada, tornando possível a realização deste trabalho.

À minha família, pelo amor, apoio, incentivo e confiança, fundamentais para que eu pudesse superar os desafios encontrados durante essa caminhada acadêmica.

A todos os professores, pelos conhecimentos compartilhados, pela dedicação ao ensino e pela importante contribuição para minha formação como engenheiro.

Em especial, ao professor Aloísio, pela orientação, confiança e pelas inúmeras oportunidades proporcionadas ao longo da graduação, especialmente na monitoria e na iniciação científica, experiências que foram essenciais para meu crescimento acadêmico e profissional.

À minha namorada, Laura, pelo carinho, compreensão, incentivo e por estar ao meu lado durante todos os momentos desta trajetória.

Ao meu primo Samuel, pelo companheirismo, pela convivência durante a graduação e por toda a ajuda na superação dos desafios encontrados ao longo dessa trajetória.

Aos meus amigos Caio, Bernardo, Carol, Eustáquio, Leandro, Rodrigo, Thales e Turbano, pela amizade, companheirismo e pelos inúmeros momentos compartilhados ao longo da graduação, tornando essa caminhada mais leve e especial.

Ao meu eterno amigo Humberto Cerceau, que, mesmo não estando mais presente, deixou lembranças, ensinamentos e uma amizade que levarei para toda a vida.

Por fim, agradeço a todos que, de alguma forma, contribuíram para a realização deste trabalho e para minha formação pessoal e profissional.

RESUMO

O problema de máquinas paralelas idênticas com restrição de recursos consiste em determinar a alocação e o sequenciamento de tarefas em máquinas paralelas, considerando que tarefas que compartilham um mesmo recurso não podem ser processadas simultaneamente. Trata-se de um problema NP-difícil, cuja elevada complexidade computacional dificulta a obtenção de soluções ótimas por métodos exatos em instâncias de maior porte. Nesse contexto, este trabalho teve como objetivo implementar, calibrar e comparar as meta-heurísticas *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing (SA)* para a resolução desse problema, considerando a minimização do *makespan*. Inicialmente, foi realizada a calibração dos parâmetros por meio de planejamento de experimentos. Em seguida, os algoritmos foram avaliados utilizando instâncias da literatura e um conjunto de instâncias geradas especificamente para este trabalho, contemplando diferentes quantidades de tarefas, máquinas e recursos compartilhados. Os resultados demonstraram que todas as meta-heurísticas produziram soluções de elevada qualidade, alcançando o limite inferior em diversas instâncias e apresentando baixos desvios em relação a esse limite nos demais casos. Na comparação com um modelo de Programação Linear Inteira Mista resolvido pelo *Gurobi*, as meta-heurísticas encontraram soluções equivalentes para as instâncias de pequeno porte em tempos computacionais significativamente inferiores. Entre os métodos avaliados, o *Simulated Annealing* apresentou o melhor desempenho global, destacando-se pelos menores desvios percentuais médios em relação ao limite inferior, pelos menores tempos computacionais e pela menor variabilidade dos resultados entre as 30 execuções, especialmente nas instâncias de médio e grande porte.

Palavras-chave: Sequenciamento da produção; Máquinas paralelas; Meta-heurísticas; Recursos compartilhados; *Makespan*.

ABSTRACT

The identical parallel-machine scheduling problem with resource constraints consists of determining the assignment and sequencing of jobs on parallel machines, considering that jobs sharing the same resource cannot be processed simultaneously. This is an NP-hard problem whose high computational complexity makes obtaining optimal solutions through exact methods difficult for larger instances. In this context, this study aimed to implement, calibrate, and compare the Greedy Randomized Adaptive Search Procedure (GRASP), Iterated Local Search (ILS), Variable Neighborhood Search (VNS), and Simulated Annealing (SA) metaheuristics to solve this problem, considering the minimization of the makespan. Initially, the parameters were calibrated using a design of experiments. Subsequently, the algorithms were evaluated using benchmark instances from the literature and a set of instances specifically generated for this study, comprising different numbers of jobs, machines, and shared resources. The results showed that all the metaheuristics were capable of producing high-quality solutions, obtaining optimal or near-optimal solutions for the evaluated instances. Compared with a Mixed-Integer Linear Programming model solved using Gurobi, the metaheuristics obtained equivalent solutions for the small-scale instances in significantly shorter computational times. Among the evaluated methods, Simulated Annealing achieved the best overall performance, standing out for its lower mean percentage deviations from the lower bound, shorter computational times, and lower variability across the 30 runs, particularly for medium- and large-scale instances.

Keywords: Production scheduling; Parallel machines; Metaheuristics; Shared resources; Makespan.

LISTA DE FIGURAS

Figura 1 - Exemplo de gráfico de <i>Gantt</i> para sequenciamento em múltiplos recursos...	7
Figura 2 - Representação do ambiente de máquinas paralelas.....	10
Figura 3 - Estrutura de precedência de uma tarefa em um sistema <i>flow shop</i>	13
Figura 4 - Representação do fluxo de tarefas em uma máquina no ambiente <i>job shop</i>	14
Figura 5 - Pseudocódigo do <i>GRASP</i>	20
Figura 6 - Pseudocódigo do <i>ILS</i>	22
Figura 7 - Pseudocódigo do <i>VNS</i>	24
Figura 8 - Pseudocódigo do <i>SA</i>	27
Figura 9 - Representação da solução utilizada pelas meta-heurísticas.....	34
Figura 10 - Exemplos das estruturas de vizinhança utilizadas.....	35
Figura 11 - Tempo computacional médio das meta-heurísticas em função do número de tarefas das instâncias de Chung.....	49
Figura 12 - Estrutura da instância 20-4-4.....	63
Figura 13 - Gráfico de <i>Gantt</i> da solução obtida pelo <i>GRASP</i> para a instância 20-4-4.....	64
Figura 14 - Gráfico de <i>Gantt</i> da solução obtida pelo <i>ILS</i> para a instância 20-4-4.....	64
Figura 15 - Gráfico de <i>Gantt</i> da solução obtida pelo <i>VNS</i> para a instância 20-4-4.....	65
Figura 16 - Gráfico de <i>Gantt</i> da solução obtida pelo <i>Simulated Annealing</i> para a instância 20-4-4.....	66

LISTA DE TABELAS

Tabela 1 - Instâncias de teste utilizadas na definição dos parâmetros.....	39
Tabela 2 - Parâmetros definidos para o <i>GRASP</i>	41
Tabela 3 - Parâmetros definidos para o <i>ILS</i>	42
Tabela 4 - Parâmetros definidos para o <i>VNS</i>	43
Tabela 5 - Parâmetros definidos para o <i>SA</i>	45
Tabela 6 - Parâmetros definidos para as meta-heurísticas.....	45
Tabela 7 - Desvio percentual médio das meta-heurísticas nas instâncias da literatura.....	48
Tabela 8 - Tempo computacional médio das meta-heurísticas por porte de instância.....	48
Tabela 9 - Configuração das instâncias geradas para o trabalho.....	50
Tabela 10 - Desvio percentual médio das meta-heurísticas para as instâncias de 20 tarefas.....	51
Tabela 11 - Tempo computacional médio em (s) das meta-heurísticas para as instâncias de 20 tarefas.....	52
Tabela 12 - Desvio-padrão do <i>makespan</i> das meta-heurísticas para as instâncias de 20 tarefas.....	53
Tabela 13 - Desvio percentual médio obtido pelas meta-heurísticas para as instâncias de 50 tarefas.....	53
Tabela 14 - Tempo computacional médio (s) das meta-heurísticas para as instâncias de 50 tarefas.....	54
Tabela 15 - Desvio-padrão do desvio absoluto das meta-heurísticas para as instâncias de 50 tarefas.....	55
Tabela 16 - Desvio percentual médio das meta-heurísticas para as instâncias de 100 e 150 tarefas.....	56
Tabela 17 - Tempo computacional médio em (s) das meta-heurísticas para as instâncias de 100 e 150 tarefas.....	57
Tabela 18 - Desvio-padrão do desvio absoluto das meta-heurísticas para as instâncias de 100 e 150 tarefas.....	58
Tabela 19 - Resumo dos desvios percentuais nas replicações da instância 150-2-2.....	58
Tabela 20 - Comparação entre as soluções obtidas pelo Gurobi e pelas meta-heurísticas para as instâncias de 20 tarefas.....	62

Tabela 21 - Tempo computacional em (s) do <i>Gurobi</i> e das meta-heurísticas para as instâncias de 20 tarefas.....	63
---	----

SUMÁRIO

1	INTRODUÇÃO	1
1.1	Objetivos.....	3
1.1.1	Objetivo geral	3
1.1.2	Objetivos específicos	3
2	PROBLEMA DE SEQUENCIAMENTO DA PRODUÇÃO	4
2.1	Sequenciamento da produção	4
2.1.1	Medidas de desempenho.....	5
2.1.2	Ambientes de sequenciamento	7
2.1.3	Classificação dos problemas de sequenciamento	8
2.1.3.1	Máquina única	8
2.1.3.2	Máquinas paralelas	9
2.1.3.3	Máquinas em série ou fluxo (flow shop)	12
2.1.3.4	Oficina de máquinas (job shop).....	13
3	MÉTODOS DE SOLUÇÃO PARA PROBLEMAS DE SEQUENCIAMENTO	15
3.1	Métodos exatos (modelo matemático, programação inteira).....	15
3.2	Heurísticas	16
3.3	Meta-heurísticas.....	18
3.3.1	<i>Greedy Randomized Adaptive Search Procedure (GRASP)</i>	19
3.3.2	<i>Iterated Local Search (ILS)</i>	21
3.3.3	<i>Variable Neighborhood Search (VNS)</i>	22
3.3.4	<i>Simulated Annealing (SA)</i>	25
4	ESTUDOS E APLICAÇÕES DO PROBLEMA DE MÁQUINAS PARALELAS IDÊNTICAS COM RESTRIÇÃO DE RECURSOS	29
5	METODOLOGIA	32
5.1	Caracterização do problema	32
5.2	Função de avaliação.....	33
5.3	Representação da solução	33
5.4	Estruturas de vizinhanças	34
5.5	Solução inicial	35
5.6	Implementação dos algoritmos.....	36
5.7	Parâmetros a serem avaliados.....	37
5.7.1	Análise e definição dos parâmetros	38

5.7.2	Análise e definição dos parâmetros do <i>GRASP</i>	40
5.7.3	Análise e definição dos parâmetros do <i>ILS</i>	41
5.7.4	Análise e definição dos parâmetros do <i>VNS</i>	42
5.7.5	Análise e definição dos parâmetros do <i>Simulated Annealing</i>	44
5.7.6	Parâmetros adotados nos experimentos computacionais	45
6	RESULTADOS E DISCUSSÃO	46
6.1	Testes com instâncias da literatura	47
6.2	Testes com instâncias geradas para o trabalho	49
6.2.1	Geração das instâncias	50
6.2.2	Análise das instâncias de pequeno porte	51
6.2.3	Análise das instâncias de médio porte	53
6.2.4	Análise das instâncias de grande porte	55
6.2.5	Análise das replicações da instância 150-2-2	58
6.3	Comparação com o modelo de programação linear resolvido pelo <i>Gurobi</i>	59
6.4	Exemplo de soluções obtidas pelas abordagens propostas	63
7	CONCLUSÃO	67
	REFERÊNCIAS BIBLIOGRÁFICAS	69
	APÊNDICE	72

1 INTRODUÇÃO

O planejamento e controle da produção desempenha papel fundamental na gestão dos sistemas produtivos, contribuindo para a utilização eficiente dos recursos disponíveis, o cumprimento dos prazos de entrega e a melhoria do desempenho operacional das organizações. Dentro desse contexto, as decisões relacionadas à programação da produção assumem grande importância, uma vez que determinam como os recursos serão utilizados ao longo do tempo para atender à demanda existente.

Segundo Arenales *et al.* (2007), as decisões relacionadas à produção podem ser organizadas em níveis estratégico, tático e operacional. No nível operacional, destacam-se as atividades de programação da produção, responsáveis pela designação das tarefas aos recursos disponíveis e pela definição da sequência de execução dessas atividades. Tais decisões estão diretamente associadas aos problemas de sequenciamento da produção, amplamente estudados na área de Pesquisa Operacional devido à sua relevância prática e à sua elevada complexidade.

Entre os diferentes ambientes de sequenciamento encontrados na literatura, os sistemas de máquinas paralelas destacam-se por sua ampla presença em aplicações industriais e de serviços. Nesses ambientes, diversas tarefas devem ser distribuídas entre recursos capazes de operar simultaneamente, exigindo decisões relacionadas tanto à alocação das tarefas quanto à definição da ordem de processamento em cada máquina (PINEDO, 2016).

Entretanto, situações reais frequentemente apresentam restrições adicionais que tornam o processo de programação mais complexo. Em muitos sistemas produtivos, além da disponibilidade das máquinas, determinadas tarefas necessitam da utilização de recursos complementares, como operadores, ferramentas, moldes, robôs industriais ou veículos guiados automaticamente. Como esses recursos geralmente possuem disponibilidade limitada, sua utilização deve ser considerada simultaneamente às decisões de sequenciamento (VENTURA; KIM, 2003).

Esse contexto caracteriza o problema de máquinas paralelas idênticas com restrição de recursos (PMPIRR), no qual uma tarefa somente pode ser processada quando houver, ao mesmo tempo, uma máquina disponível e o recurso adicional necessário para sua execução. Como consequência, tarefas que compartilham um mesmo recurso limitado não podem ser executadas simultaneamente, mesmo quando existirem máquinas disponíveis para seu processamento (CHUNG *et al.*, 2019).

A relevância desse problema pode ser observada em diferentes setores industriais. Chung *et al.* (2019), por exemplo, estudaram uma aplicação na indústria de semicondutores, na qual diferentes tarefas compartilham moldes utilizados na fabricação de *wafers*. Como esses moldes possuem elevado custo de aquisição, sua disponibilidade é limitada, criando restrições adicionais ao processo de programação da produção. De forma semelhante, Soares e Carvalho (2020) abordaram aplicações em sistemas flexíveis de manufatura, nos quais diferentes tarefas compartilham ferramentas necessárias ao processamento das operações.

Além de sua importância prática, o problema de máquinas paralelas com restrição de recursos apresenta elevada complexidade computacional. Variantes desse problema pertencem à classe NP-difícil, o que dificulta a obtenção de soluções ótimas por métodos exatos, especialmente à medida que aumentam o número de tarefas, recursos e restrições consideradas (CHUNG *et al.*, 2019; SOARES; CARVALHO, 2020). Dessa forma, torna-se necessário o desenvolvimento de métodos capazes de produzir soluções de elevada qualidade em tempos computacionais compatíveis com a tomada de decisão em ambientes produtivos reais.

Nesse cenário, as meta-heurísticas têm se destacado como importantes ferramentas para a resolução de problemas complexos de otimização combinatória. Métodos como *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing (SA)* vêm sendo amplamente empregados na literatura devido à sua capacidade de explorar grandes espaços de soluções e obter resultados competitivos em tempos computacionais reduzidos (TALBI, 2009; LOPES; RODRIGUES; STEINER, 2013).

Diante desse contexto, formula-se a seguinte pergunta de pesquisa: entre as meta-heurísticas *GRASP*, *ILS*, *VNS* e *Simulated Annealing*, qual apresenta o melhor equilíbrio entre qualidade das soluções, tempo computacional e estabilidade dos resultados nas diferentes configurações do problema de máquinas paralelas idênticas com restrição de recursos? Para responder a essa questão, este trabalho implementa, calibra e compara essas quatro meta-heurísticas, considerando diferentes conjuntos de instâncias e a minimização do *makespan*.

1.1 Objetivos

1.1.1 Objetivo geral

O objetivo geral deste trabalho é implementar, calibrar e comparar as meta-heurísticas *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing (SA)* para a resolução do problema de máquinas paralelas idênticas com restrição de recursos, considerando a minimização do *makespan* e avaliando a qualidade das soluções, o tempo computacional e a estabilidade dos resultados.

1.1.2 Objetivos específicos

Para alcançar o objetivo geral, foram estabelecidos os seguintes objetivos específicos, que orientam as etapas de investigação, implementação, calibração e avaliação dos métodos estudados:

- Investigar a aplicação de meta-heurísticas na resolução do problema de máquinas paralelas idênticas com restrição de recursos;
- Implementar as meta-heurísticas *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing (SA)* para a resolução do problema estudado;
- Calibrar os parâmetros das meta-heurísticas por meio de planejamento de experimentos;
- Comparar a qualidade das soluções e o desempenho computacional das meta-heurísticas implementadas;
- Verificar a capacidade das meta-heurísticas em obter soluções equivalentes ou próximas às encontradas por um modelo de Programação Linear Inteira Mista para instâncias de pequeno porte;
- Avaliar o comportamento das meta-heurísticas em instâncias de diferentes portes e níveis de complexidade;
- Identificar a meta-heurística que apresenta o melhor equilíbrio entre qualidade das soluções, tempo computacional e estabilidade dos resultados.

2 PROBLEMA DE SEQUENCIAMENTO DA PRODUÇÃO

2.1 Sequenciamento da produção

O sequenciamento é uma das atividades do planejamento e controle da produção (PCP) responsável por determinar a ordem em que as tarefas, ordens ou trabalhos (*jobs*) serão processados nos recursos disponíveis. Essa atividade tem como objetivo otimizar o fluxo de produção, promovendo uma utilização mais eficiente dos recursos e contribuindo para a melhoria do desempenho do sistema produtivo (SLACK; BRANDON-JONES; BURGESS, 2023).

Desse modo, essa atividade possui papel fundamental na gestão dos sistemas de produção. Segundo Tubino (2017), uma organização adequada das tarefas permite reduzir tempos de espera entre operações, minimizar atrasos no processamento e melhorar a utilização dos recursos disponíveis, contribuindo para um fluxo de produção mais eficiente e organizado. No entanto, na prática, muitos desses problemas de sequenciamento são resolvidos de maneira intuitiva, frequentemente seguindo regras de prioridade, como a ordem de chegada das tarefas, o que nem sempre resulta em soluções eficientes, principalmente em ambientes produtivos mais complexos (CONWAY; MAXWELL; MILLER, 1967).

Adicionalmente, a obtenção de soluções ótimas pode ser limitada pela complexidade dos problemas de sequenciamento, especialmente à medida que aumenta a dimensão dos sistemas produtivos (PINEDO, 2016).

Diante do cenário competitivo atual, as indústrias são pressionadas a cumprir os prazos de entrega prometidos aos clientes, uma vez que o não cumprimento pode gerar impactos negativos à imagem da organização. Nesse contexto, o sequenciamento da produção surge como um importante aliado, auxiliando na alocação eficiente de recursos e tarefas ao longo do tempo, com o objetivo de otimizar indicadores de desempenho do sistema produtivo (PINEDO, 2016).

Os elementos envolvidos nesse processo podem assumir diferentes formas, variando conforme o sistema produtivo em que estão inseridos. Assim, os recursos podem representar máquinas em sistemas de manufatura, pistas de aeroportos ou equipes em canteiros de obras. Já as tarefas podem corresponder a operações em processos produtivos, decolagens e pousos em aeroportos ou etapas em projetos de construção (PINEDO, 2016), evidenciando a ampla aplicabilidade dos problemas de sequenciamento em diferentes contextos produtivos e organizacionais.

2.1.1 Medidas de desempenho

Nos problemas de sequenciamento da produção, além de determinar a ordem de execução das tarefas, também é necessário definir critérios que permitam avaliar a qualidade das soluções obtidas. Esses critérios são conhecidos como medidas de desempenho ou funções objetivo, e por meio dessas medidas é possível comparar as soluções geradas e selecionar aquela que melhor atende aos objetivos do sistema (PINEDO, 2016).

Para compreender melhor essas métricas, é necessário primeiro apresentar a notação utilizada na representação dos problemas de sequenciamento. Segundo Pinedo (2016), nesses cenários considera-se que o número de tarefas e máquinas é finito, sendo representados respectivamente por n e m . Nessa notação, o índice j é utilizado para identificar cada tarefa (*job*), enquanto o índice i representa cada máquina ou recurso produtivo. Dessa forma, o par (i, j) representa a operação correspondente ao processamento da tarefa j na máquina i .

A partir dessa notação, algumas variáveis são utilizadas para representar o comportamento das tarefas no sistema produtivo (Quadro 1).

Quadro 1 – Variáveis utilizadas em problemas de sequenciamento

Símbolo	Definição
n	Número de tarefas.
m	Número de máquinas.
j	Índice associado às tarefas.
i	Índice associado às máquinas.
p_j	Tempo de processamento da tarefa j .
C_j	Instante em que a tarefa j é concluída.
C_{max}	Tempo de conclusão da última tarefa processada (<i>makespan</i>).

Fonte: Adaptado de Pinedo (2016).

A literatura apresenta diferentes medidas de desempenho para os problemas de sequenciamento. Entre elas, o tempo de fluxo representa o período durante o qual uma tarefa permanece no sistema. No modelo básico em que todas as tarefas estão disponíveis no instante inicial, o tempo de fluxo está diretamente relacionado ao instante de conclusão da tarefa. Também podem ser consideradas medidas associadas às datas de entrega, como o atraso das tarefas, que permite avaliar o cumprimento dos prazos estabelecidos (BAKER; TRIETSCH, 2019).

Embora a literatura apresente diversas medidas de desempenho para problemas de sequenciamento, o *makespan* (C_{max}) é um dos critérios mais empregados. Esse indicador

representa o instante de conclusão da última tarefa processada, refletindo diretamente a duração total do cronograma de produção e a utilização global dos recursos do sistema (PINEDO, 2016).

Contudo, é importante destacar que diferentes métricas de desempenho podem levar a soluções distintas para um mesmo problema, uma vez que cada critério prioriza um aspecto específico do sistema produtivo. Assim, a escolha da função objetivo deve estar alinhada aos objetivos da organização (PINEDO, 2016).

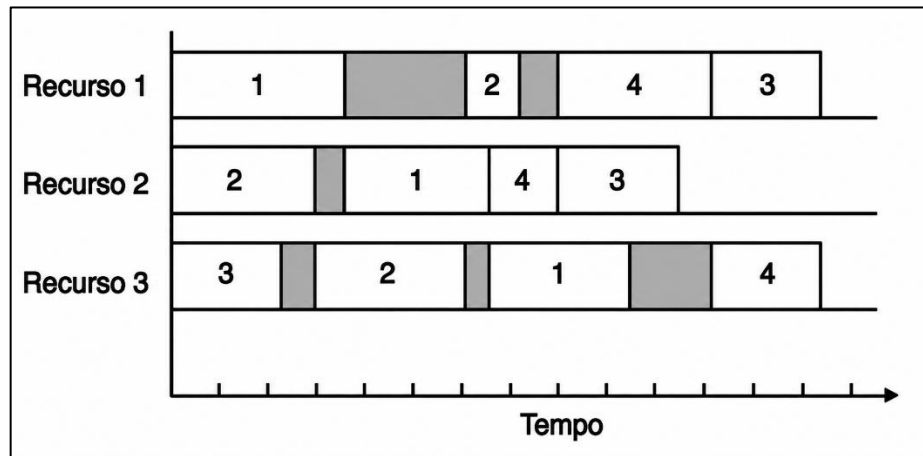
Adicionalmente, na literatura clássica de sequenciamento, observa-se que as medidas mais utilizadas incluem o tempo médio de conclusão e o *makespan*, além de medidas relacionadas ao tempo de fluxo e ao atraso das tarefas. Essas medidas permitem avaliar o desempenho das soluções de forma objetiva e são classificadas como medidas regulares, pois são definidas a partir dos tempos de conclusão das tarefas e têm como objetivo sua minimização (CONWAY; MAXWELL; MILLER, 1967).

Além dos indicadores numéricos, a representação gráfica das soluções também desempenha um papel relevante na análise dos problemas de sequenciamento. Nesse sentido, o gráfico de *Gantt* é amplamente utilizado para ilustrar a alocação das tarefas ao longo do tempo nos diferentes recursos produtivos. De acordo com Baker e Trietsch (2019), essa representação permite visualizar de forma clara a relação entre tarefas e recursos, uma vez que os recursos são apresentados no eixo vertical e o tempo no eixo horizontal.

Ainda segundo os autores, o gráfico de *Gantt* não apenas facilita a visualização do sequenciamento, mas também auxilia na análise e comparação de diferentes cronogramas, permitindo avaliar o desempenho das soluções obtidas. Desse modo, essa ferramenta contribui tanto para a compreensão do problema quanto para a avaliação dos resultados gerados.

A FIG. 1 apresenta um exemplo de sequenciamento representado por meio de um gráfico de *Gantt*.

Figura 1 – Exemplo de gráfico de *Gantt* para sequenciamento em múltiplos recursos



Fonte: Elaboração própria (adaptado de BAKER; TRIETSCH, 2019).

Na FIG. 1, observa-se a distribuição das tarefas ao longo do tempo em diferentes recursos produtivos, evidenciando a ordem de processamento e a execução simultânea das tarefas em recursos distintos. Além disso, é possível identificar intervalos de ociosidade entre operações, permitindo analisar a eficiência na utilização dos recursos e a organização do sequenciamento.

Diante disso, torna-se essencial compreender os diferentes contextos de aplicação dos problemas de sequenciamento, os quais serão abordados na próxima seção.

2.1.2 Ambientes de sequenciamento

Na literatura de sequenciamento, os ambientes são representados pelo campo α na notação $\alpha | \beta | \gamma$, descrevendo o ambiente de máquinas em que as tarefas são processadas (PINEDO, 2016). Nesse contexto, os diferentes ambientes podem ser classificados de acordo com a quantidade de recursos disponíveis e a forma como as tarefas são distribuídas e processadas nesses recursos, permitindo a identificação de distintas configurações de sistemas produtivos (BAKER; TRIETSCH, 2019).

Além do ambiente de máquinas representado pelo campo α , a notação também considera características das tarefas, descritas no campo β . Entre essas características, destaca-se a possibilidade de interrupção das tarefas durante o processamento, representada pelo parâmetro β_1 . Nesse contexto, $\beta_1 = \emptyset$ indica ausência de preempção, enquanto $\beta_1 = t\text{-pmtn}$ e $\beta_1 = \text{pmtn}$ indicam, respectivamente, preempção e preempção arbitrária (KAABI; HARRATH, 2014).

O campo γ , por sua vez, representa o critério de otimização adotado no problema. Entre as funções objetivo apresentadas por Kaabi e Harrath (2014), encontram-se a minimização do

makespan (C_{max}), do tempo total de conclusão ($\sum C_j$), do tempo total ponderado de conclusão ($\sum w_j C_j$), do *lateness* máximo (L_{max}) e do atraso máximo (T_{max}).

Segundo Pinedo (2016), a complexidade dos problemas de sequenciamento está diretamente relacionada ao ambiente considerado, isto é, configurações diferentes podem apresentar níveis distintos de dificuldade de resolução.

Dentre os principais ambientes de sequenciamento abordados na literatura, destacam-se o ambiente de máquina única, as máquinas paralelas, o *flow shop* e o *job shop*, os quais representam diferentes configurações dos sistemas produtivos e serão detalhados na sequência.

2.1.3 Classificação dos problemas de sequenciamento

Com base nos diferentes ambientes de sequenciamento apresentados anteriormente, torna-se necessário classificá-los de forma mais detalhada, uma vez que cada configuração apresenta características específicas que influenciam diretamente a complexidade do problema e as estratégias de solução adotadas. Dessa forma, os principais ambientes de sequenciamento são analisados a seguir, destacando suas principais características e aplicações.

2.1.3.1 Máquina única

Dentre os ambientes de sequenciamento, o problema de máquina única é classificado como o mais simples, sendo caracterizado pela existência de apenas um recurso disponível para o processamento das tarefas (BAKER; TRIETSCH, 2019; PINEDO, 2016). Na notação $\alpha | \beta | \gamma$, esse ambiente é representado por $\alpha = 1$, indicando a presença de apenas uma máquina no sistema (PINEDO, 2016). Diante disso, as tarefas são executadas de forma sequencial, uma vez que a disponibilidade de um único recurso impede o processamento simultâneo.

Apesar de sua simplicidade estrutural, o problema de máquina única possui grande relevância, por representar uma forma simplificada dos ambientes de sequenciamento mais complexos. Na prática, sistemas produtivos de maior porte podem ser decompostos em subproblemas de máquina única, especialmente quando há a presença de gargalos no sistema. Além disso, os resultados obtidos nesse tipo de modelo não apenas contribuem para a compreensão do próprio ambiente de máquina única, mas também servem como base para o desenvolvimento de heurísticas aplicáveis a sistemas mais complexos (PINEDO, 2016).

Segundo Baker e Trietsch (2019), além da limitação de um único recurso, esse tipo de problema é caracterizado pelas seguintes condições:

1. Todas as tarefas possuem apenas uma operação e já estão disponíveis para serem processadas no instante zero;
2. A máquina só processa uma tarefa por vez;
3. Os tempos de preparação não dependem da ordem das tarefas e já estão incluídos no tempo de processamento;
4. Todas as informações do problema são conhecidas previamente e não mudam ao longo do tempo;
5. O recurso está sempre disponível para uso, sem falhas ou interrupções.
6. A máquina não fica parada enquanto houver tarefas esperando para serem processadas;
7. Quando uma tarefa começa a ser processada, ela vai até o fim sem ser interrompida.

Do ponto de vista prático, Conway, Maxwell e Miller (1967) mostram que o modelo de máquina única é mais comum do que parece, mesmo em sistemas produtivos complexos. Isso acontece porque, em muitas situações reais, os processos são realizados de forma sequencial, ou seja, um produto é processado por vez. Esse comportamento pode ser observado, por exemplo, na produção de tintas, em que diferentes produtos são fabricados um após o outro no mesmo equipamento.

Os autores também destacam que, mesmo em sistemas com várias máquinas, pode existir um recurso dominante que determina o ritmo da produção. Nessas situações, todo o sistema passa a se comportar como se fosse uma única máquina, o que permite uma modelagem e análise mais simplificada, como ocorre na indústria de papel. Dessa forma, mesmo em ambientes aparentemente complexos, o comportamento pode ser simplificado e analisado como um problema de máquina única.

2.1.3.2 Máquinas paralelas

O problema de máquinas paralelas pode ser visto como uma extensão do problema de máquina única, pois envolve a presença de várias máquinas disponíveis simultaneamente para processar as tarefas. Esse tipo de sistema é bastante relevante tanto na teoria quanto na prática, já que é comum encontrar recursos operando em paralelo nos sistemas produtivos reais. Nesse contexto, o problema de sequenciamento não se limita apenas à determinação da ordem das tarefas, mas também à definição de em qual máquina cada tarefa será processada, tornando-o mais complexo ao envolver simultaneamente decisões de alocação e de sequenciamento (PINEDO, 2016).

Assim como no ambiente de máquina única, no ambiente de máquinas paralelas, as tarefas possuem apenas uma operação. Nesse contexto, considera-se um conjunto de tarefas

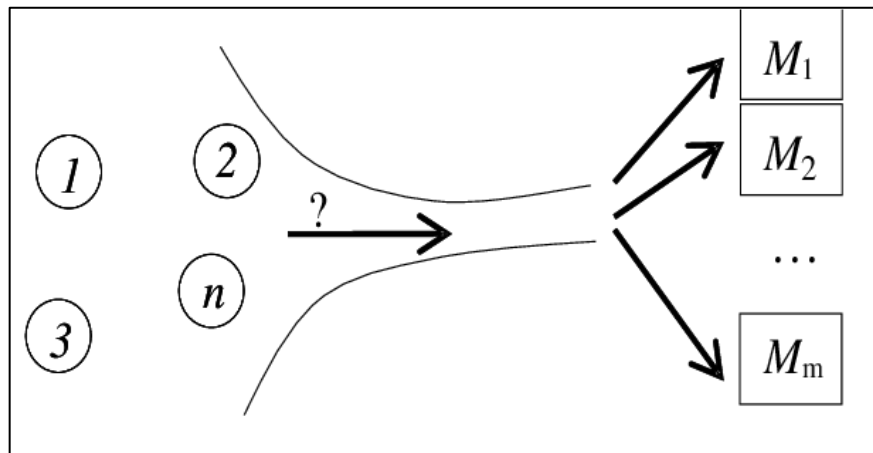
disponíveis desde o tempo inicial e múltiplas máquinas capazes de operar em paralelo, reforçando a importância da correta distribuição das tarefas entre os recursos disponíveis (BAKER; TRIETSCH, 2019).

Nessa configuração clássica, assume-se ainda que cada máquina processa apenas uma tarefa por vez e que as tarefas não sofrem interrupções durante sua execução (não-*preempção*). Adicionalmente, as tarefas são geralmente consideradas independentes entre si, não havendo relações de precedência, o que amplia a flexibilidade na definição do sequenciamento (CHUNG *et al.*, 2019).

Esse tipo de ambiente é avaliado principalmente pelo *makespan*, que assume papel central por estar diretamente relacionado ao balanceamento de carga entre as máquinas. Ao reduzir esse indicador, busca-se diminuir o tempo total de processamento e melhorar a utilização dos recursos. Assim, após a alocação das tarefas, procura-se minimizar o tempo associado à máquina mais carregada, que determina a duração total necessária para a conclusão de todas as atividades (MÜLLER; DIAS; ARAÚJO, 2002).

A FIG. 2 ilustra um exemplo de sequenciamento em máquinas paralelas, evidenciando a distribuição das tarefas entre as máquinas e o tempo total de processamento do sistema.

Figura 2 – Representação do ambiente de máquinas paralelas



Fonte: Kaabi e Harrath (2014).

Adicionalmente, os problemas de máquinas paralelas podem ser classificados de acordo com as características dos recursos. Na notação $\alpha | \beta | \gamma$, o ambiente de máquinas é representado no campo α , como P_m , Q_m e R_m (PINEDO, 2016), conforme descrito a seguir:

- Máquinas idênticas (P_m): são sistemas em que todas as máquinas são iguais, ou seja, não há diferença entre elas no processamento das tarefas. Assim, qualquer tarefa pode

ser executada em qualquer máquina. Porém, em alguns casos, podem existir restrições que limitam certas tarefas a máquinas específicas.

- Máquinas uniformes (Q_m): Nesse tipo de ambiente, as máquinas possuem velocidades diferentes. Isso significa que uma mesma tarefa pode levar mais ou menos tempo dependendo da máquina em que for processada. Quanto maior a velocidade da máquina, menor será o tempo de processamento.
- Máquinas não relacionadas (R_m): É uma generalização do caso anterior, no qual o tempo de processamento de uma tarefa depende tanto da máquina quanto da tarefa, ou seja, uma mesma tarefa pode ter tempos completamente diferentes em cada máquina, sem um padrão fixo.

Na prática, esses diferentes tipos de máquinas podem ser observados em diversos contextos produtivos. As máquinas idênticas, por exemplo, são comuns quando uma empresa replica equipamentos para aumentar a capacidade de um gargalo, garantindo o mesmo desempenho entre os recursos. Já as máquinas uniformes surgem quando novos equipamentos são incorporados ao sistema, muitas vezes com tecnologias diferentes, o que resulta em variações de velocidade. Por fim, as máquinas não relacionadas aparecem em ambientes mais flexíveis, nos quais cada máquina pode apresentar melhor desempenho para determinados tipos de tarefas, sem uma relação fixa entre elas (MÜLLER; DIAS; ARAÚJO, 2002).

Nesse contexto, os problemas de máquinas paralelas, sem restrições adicionais e com o objetivo de minimizar o *makespan* são representados como $P \parallel C_{max}$, para máquinas idênticas; $Q \parallel C_{max}$, para máquinas uniformes; e $R \parallel C_{max}$, para máquinas não relacionadas (LAWLER *et al.*, 1989).

Adicionalmente, destaca-se que os sistemas de máquinas paralelas são amplamente encontrados em aplicações reais, sendo utilizados em ambientes produtivos que demandam alta capacidade e processamento simultâneo de tarefas, o que reforça a relevância do estudo desses problemas tanto do ponto de vista teórico quanto prático (CHUNG *et al.*, 2019).

Segundo Müller, Dias e Araújo (2002), a resolução de problemas de máquinas paralelas é, em grande parte, baseada no uso de métodos heurísticos. Os autores destacam que diferentes abordagens são empregadas na literatura, incluindo métodos para construção de soluções iniciais e estratégias de melhoria, aplicados tanto em máquinas paralelas idênticas quanto em ambientes mais complexos, como máquinas uniformes e não relacionadas.

Em problemas de máquinas paralelas, aplicações industriais reais frequentemente apresentam limitações adicionais relacionadas à disponibilidade de recursos compartilhados necessários ao processamento das tarefas. Essas restrições podem estar associadas a diferentes elementos do sistema produtivo, exigindo que, além da alocação das tarefas entre as máquinas disponíveis, seja considerada também a disponibilidade dos recursos requeridos para a execução das atividades (CHUNG *et al.*, 2019).

Em aplicações industriais, esses recursos adicionais podem assumir diferentes formas, como operadores, ferramentas, robôs industriais, veículos guiados automaticamente (*Automated Guided Vehicles – AGVs*), dispositivos de fixação e outros recursos auxiliares necessários ao processamento das tarefas. A indisponibilidade desses recursos pode impedir a execução de determinadas atividades, mesmo quando houver máquinas disponíveis no sistema produtivo (VENTURA; KIM, 2003).

Problemas dessa natureza são classificados como problemas de sequenciamento em máquinas paralelas com restrição de recursos, nos quais recursos adicionais atuam como restrições ao sistema produtivo e influenciam diretamente o processo de programação das tarefas (CHUNG *et al.*, 2019).

Além disso, problemas de máquinas paralelas com restrições de recursos exigem que as decisões de sequenciamento sejam tomadas considerando simultaneamente a disponibilidade das máquinas e dos recursos compartilhados. Dessa forma, restrições adicionais introduzem novas condições ao problema, aumentando a dificuldade de obtenção de soluções viáveis e ampliando a complexidade do processo de programação da produção (VENTURA; KIM, 2003).

A consideração simultânea da disponibilidade das máquinas e dos recursos compartilhados aumenta a complexidade do processo de sequenciamento. Segundo Chung *et al.* (2019) e Soares e Carvalho (2020), variantes desse problema pertencem à classe NP-difícil, o que dificulta a obtenção de soluções ótimas por métodos exatos, especialmente para instâncias de maior porte. Nesse contexto, métodos heurísticos e meta-heurísticos têm sido empregados como alternativas para a obtenção de soluções de elevada qualidade em tempos computacionais viáveis.

2.1.3.3 Máquinas em série ou fluxo (*flow shop*)

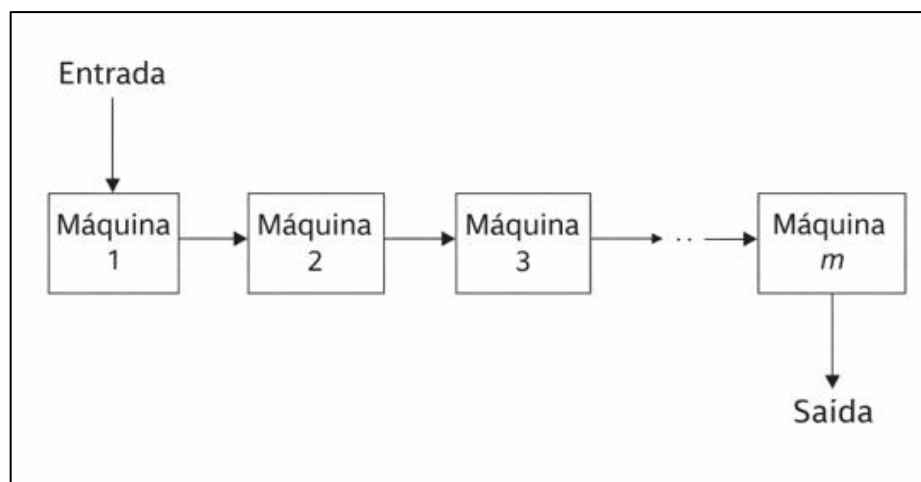
O ambiente de máquinas em série, conhecido como *flow shop*, caracteriza-se por um sistema em que as máquinas estão organizadas em sequência e as tarefas percorrem esse fluxo seguindo

uma mesma ordem de processamento. Nesse contexto, cada tarefa é composta por operações realizadas em máquinas distintas, respeitando uma rota previamente definida (BAKER; TRIETSCH, 2019).

No modelo clássico de *flow shop*, todas as tarefas percorrem as máquinas na mesma sequência. Além disso, as operações não podem ser interrompidas e cada máquina processa apenas uma tarefa por vez. Esse ambiente é amplamente utilizado para representar linhas de produção com fluxo contínuo, como as encontradas nas indústrias automobilística, alimentícia e de bens de consumo (BAKER; TRIETSCH, 2019).

A FIG. 3 apresenta uma representação simplificada do fluxo de processamento em um ambiente *flow shop*.

Figura 3 – Estrutura de precedência de uma tarefa em um sistema *flow shop*



Fonte: Elaboração própria (adaptado de BAKER; TRIETSCH, 2019).

À medida que o número de máquinas aumenta, o problema torna-se computacionalmente mais complexo, sendo classificado como NP-difícil para três ou mais máquinas quando o objetivo é minimizar o *makespan* (GAREY; JOHNSON; SETHI, 1976). Dessa forma, heurísticas e meta-heurísticas são frequentemente utilizadas para obtenção de soluções de boa qualidade em tempos computacionais adequados (SAYADI; RAMEZANIAN; GHAFARI-NASAB, 2010).

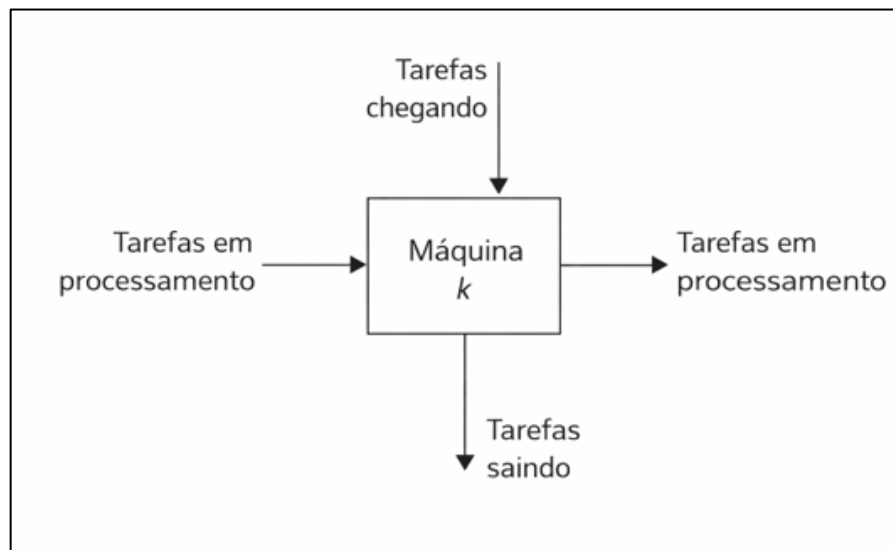
2.1.3.4 Oficina de máquinas (*job shop*)

No problema de *job shop*, considera-se um conjunto de tarefas composto por diversas operações que devem ser processadas em máquinas específicas. Cada operação possui um tempo de processamento definido e deve respeitar relações de precedência previamente estabelecidas (GONÇALVES; MENDES; RESENDE, 2002).

Diferentemente do ambiente *flow shop*, no qual todas as tarefas seguem a mesma sequência de máquinas, no ambiente *job shop* cada tarefa pode possuir um roteiro próprio de processamento. Essa característica proporciona maior flexibilidade operacional, mas também aumenta significativamente a complexidade do problema (BAKER; TRIETSCH, 2019).

A FIG. 4 apresenta uma representação simplificada do fluxo de tarefas em um ambiente *job shop*.

Figura 4 – Representação do fluxo de tarefas em uma máquina no ambiente *job shop*



Fonte: Elaboração própria (adaptado de BAKER; TRIETSCH, 2019).

Devido a essa elevada flexibilidade, o problema de *job shop* é amplamente reconhecido como um dos ambientes mais complexos da literatura de sequenciamento (BAKER; TRIETSCH, 2019).

Diante do exposto, verifica-se que os diferentes ambientes de sequenciamento apresentam graus distintos de complexidade, exigindo abordagens específicas para sua modelagem e resolução. Essa diversidade de configurações evidencia a importância do estudo de métodos de solução adequados, os quais serão abordados nas próximas seções.

3 MÉTODOS DE SOLUÇÃO PARA PROBLEMAS DE SEQUENCIAMENTO

Segundo Pinedo (2016), problemas de sequenciamento podem apresentar elevada complexidade computacional devido às dificuldades encontradas em problemas de otimização combinatória. Dessa forma, diferentes métodos são utilizados para sua resolução, sendo a escolha da abordagem influenciada pelas características do sistema produtivo, pelas restrições existentes e pelos objetivos definidos.

Em problemas mais simples, determinados métodos podem fornecer soluções ótimas com desempenho computacional satisfatório. Entretanto, conforme o porte do problema aumenta e novas restrições são adicionadas ao modelo, o esforço computacional necessário para encontrar a melhor solução tende a crescer significativamente (PINEDO, 2016). Dessa forma, diferentes abordagens passaram a ser desenvolvidas e aplicadas na literatura, buscando equilibrar qualidade das soluções e viabilidade computacional.

Entre os métodos encontrados na literatura destacam-se os métodos exatos, as heurísticas e as meta-heurísticas. Cada abordagem apresenta características específicas e sua aplicação depende do tamanho do problema, das restrições existentes e do objetivo considerado. Estudos envolvendo problemas de sequenciamento em máquinas paralelas mostram a utilização dessas diferentes abordagens, incluindo métodos exatos, heurísticos e meta-heurísticos aplicados em diversos contextos de produção (CHUNG *et al.*, 2019).

Nas próximas seções são apresentados os principais métodos utilizados na resolução de problemas de sequenciamento, destacando suas principais características e aplicações.

3.1 Métodos exatos (modelo matemático, programação inteira)

Os métodos exatos têm como objetivo encontrar a solução ótima de um problema por meio da construção de modelos matemáticos capazes de representar o comportamento do sistema estudado. Nesses modelos, o problema é descrito por meio de variáveis de decisão, funções objetivo e restrições que representam as condições operacionais do sistema (HILLIER; LIEBERMAN, 2013).

Essas abordagens são amplamente utilizadas em problemas de sequenciamento, uma vez que permitem representar a alocação de recursos às tarefas ao longo do tempo, considerando os objetivos estabelecidos para o sistema (PINEDO, 2016). Dessa forma, torna-se possível analisar diferentes alternativas de solução e identificar aquela que apresenta melhor desempenho de acordo com os critérios definidos.

Entre as principais técnicas utilizadas destacam-se a Programação Linear Inteira, Programação Dinâmica e métodos de enumeração implícita, como o *Branch and Bound*. Na Programação Inteira, as variáveis de decisão assumem valores inteiros, frequentemente binários, permitindo representar decisões relacionadas à ordem das tarefas, à alocação de recursos e às restrições existentes no sistema produtivo (WOLSEY, 1998).

De maneira geral, os modelos matemáticos aplicados aos problemas de sequenciamento são compostos por uma função objetivo, como minimização do *makespan*, atraso total ou tempo médio de fluxo, e por restrições que representam as características e condições do problema analisado (BAKER; TRIETSCH, 2019).

Na prática, modelos matemáticos são frequentemente implementados por meio de softwares de otimização que permitem estruturar parâmetros, variáveis de decisão, funções objetivo e restrições de maneira organizada. Entre essas ferramentas destaca-se o *Gurobi Optimizer*, amplamente utilizado na resolução de problemas de Pesquisa Operacional devido à sua elevada eficiência computacional na solução de modelos de Programação Linear, Programação Linear Inteira e Programação Linear Inteira Mista. O *solver* emprega algoritmos avançados de otimização, como *Branch and Bound*, *Branch and Cut* e técnicas de geração de cortes, permitindo a obtenção de soluções ótimas ou limites de otimalidade para problemas de diferentes níveis de complexidade (GUROBI OPTIMIZATION, LLC, 2025).

Embora os métodos exatos apresentem a vantagem de produzir soluções ótimas, o aumento do número de tarefas, recursos e restrições tende a elevar significativamente o esforço computacional necessário para resolução do problema. Dessa forma, em problemas de maior porte, a aplicação dessas abordagens pode tornar-se limitada, justificando a utilização de outros métodos para obtenção de soluções em tempos computacionais mais adequados (PINEDO, 2016).

3.2 Heurísticas

As heurísticas são métodos utilizados para obtenção de soluções em problemas de sequenciamento. Essas abordagens possuem grande importância principalmente em problemas mais complexos, nos quais a obtenção de soluções ótimas por métodos exatos pode demandar elevado esforço computacional (PINEDO, 2016).

Segundo Conway, Maxwell e Miller (1967), muitos problemas de sequenciamento são frequentemente resolvidos por meio de regras de prioridade utilizadas para definir a ordem de

processamento das tarefas. Essas abordagens permitem gerar soluções de forma mais simples e com menor custo computacional.

Entre as heurísticas encontradas na literatura destaca-se a *LPT (Longest Processing Time)*, utilizada para construção de soluções iniciais em problemas de máquinas paralelas. Essa heurística prioriza tarefas com maiores tempos de processamento, buscando distribuí-las entre os recursos disponíveis de forma a reduzir o desbalanceamento da carga entre as máquinas (PINEDO, 2016).

Além das heurísticas construtivas, também são utilizados métodos de busca local, cujo objetivo consiste em melhorar uma solução inicial por meio da exploração de soluções vizinhas. Nesses métodos, parte-se de uma solução existente e são realizadas modificações em sua estrutura, gerando novas soluções candidatas que podem apresentar melhor desempenho em relação à solução atual (BAKER; TRIETSCH, 2019).

Ainda segundo os autores, métodos de busca local exploram soluções próximas da solução corrente com o objetivo de identificar melhorias sucessivas no desempenho do problema analisado.

Entre os movimentos frequentemente utilizados destacam-se:

- Troca (*swap*): consiste na troca de posição entre duas tarefas na sequência;
- Realocação (*relocation*): consiste na remoção de uma tarefa de sua posição atual e inserção em uma nova posição da sequência;
- *2-opt*: consiste na inversão da ordem de uma subsequência de tarefas.

Outro procedimento frequentemente empregado é o *VND (Variable Neighborhood Descent)*, que utiliza diferentes estruturas de vizinhança de forma sequencial, permitindo explorar diferentes regiões do espaço de soluções e intensificar a busca por soluções de melhor qualidade (MLADENOVIC; HANSEN, 1997).

Dessa forma, as heurísticas podem atuar tanto na geração de soluções iniciais quanto no refinamento de soluções ao longo do processo de busca, sendo frequentemente utilizadas como componentes de métodos mais avançados para resolução de problemas de otimização combinatória (PINEDO, 2016). Nesse contexto, procedimentos heurísticos são amplamente incorporados em meta-heurísticas, contribuindo para a construção inicial das soluções e para a intensificação do processo de busca.

3.3 Meta-heurísticas

As meta-heurísticas são métodos de solução utilizados para resolver problemas de otimização de elevada complexidade, especialmente quando os métodos exatos apresentam alto custo computacional ou se tornam inviáveis para instâncias de maior porte. Diferentemente das heurísticas tradicionais, que geralmente são desenvolvidas para problemas específicos, as meta-heurísticas possuem estruturas mais gerais de busca, podendo ser adaptadas a diferentes tipos de problemas combinatórios (TALBI, 2009).

Segundo Lopes, Rodrigues e Steiner (2013), muitos problemas reais de Pesquisa Operacional apresentam elevada complexidade computacional, o que dificulta a obtenção de soluções ótimas em tempo viável. Nesse contexto, as meta-heurísticas surgem como alternativas capazes de produzir soluções de boa qualidade, mesmo sem garantir necessariamente a otimalidade.

De forma geral, as meta-heurísticas buscam equilibrar dois aspectos importantes: a intensificação e a diversificação. A intensificação está relacionada à exploração de regiões promissoras do espaço de soluções, enquanto a diversificação permite explorar diferentes regiões, evitando que o método fique preso em ótimos locais. Esse equilíbrio é essencial para melhorar a qualidade das soluções encontradas ao longo do processo de busca (TALBI, 2009).

Em problemas de sequenciamento da produção, o uso de meta-heurísticas é bastante comum devido à natureza combinatória desses problemas e ao grande número de possibilidades de ordenação das tarefas. Nesse contexto, métodos como *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing (SA)* podem ser utilizados para construir, melhorar e explorar soluções de forma eficiente (PINEDO, 2016; TALBI, 2009).

Dessa forma, as meta-heurísticas representam uma importante alternativa para a resolução de problemas de sequenciamento, permitindo obter soluções satisfatórias em tempos computacionais viáveis (PINEDO, 2016).

Após a apresentação dos conceitos gerais relacionados às meta-heurísticas, as subseções seguintes descrevem os métodos utilizados na resolução do problema de máquinas paralelas idênticas com restrição de recursos. Foram implementados o *Greedy Randomized Adaptive Search Procedure (GRASP)*, o *Iterated Local Search (ILS)*, o *Variable Neighborhood Search (VNS)* e o *Simulated Annealing (SA)*.

3.3.1 *Greedy Randomized Adaptive Search Procedure (GRASP)*

O *Greedy Randomized Adaptive Search Procedure (GRASP)* é uma meta-heurística proposta por Feo e Resende (1995), amplamente utilizada na resolução de problemas de otimização combinatória. O método é baseado em um processo iterativo composto por duas etapas principais: construção de uma solução inicial e busca local para refinamento da solução obtida.

Na fase construtiva, a solução é gerada de forma gulosa e randomizada. Inicialmente, é criada uma Lista Restrita de Candidatos (LRC), composta pelos melhores elementos candidatos segundo um critério guloso definido para o problema. Em seguida, um dos elementos da lista é selecionado aleatoriamente e incorporado à solução parcial. Esse procedimento é repetido até que uma solução viável seja construída (FEO; RESENDE, 1995).

Segundo Resende e Ribeiro (2016), o componente guloso direciona a busca para soluções promissoras, enquanto o componente aleatório permite gerar soluções diferentes ao longo das iterações, aumentando a diversidade da busca e reduzindo a chance de aprisionamento em ótimos locais.

Após a construção da solução inicial, aplica-se uma etapa de busca local com o objetivo de melhorar a solução obtida. Nessa etapa, são exploradas soluções vizinhas por meio de movimentos definidos para o problema, como troca de tarefas, realocação e inversão de subsequências. O processo continua até que nenhuma melhoria adicional seja encontrada (RESENDE; RIBEIRO, 2016).

De forma geral, o funcionamento do *GRASP* consiste na repetição sucessiva das fases de construção e busca local. A cada iteração, uma nova solução inicial é construída, permitindo explorar diferentes regiões do espaço de soluções. Ao final do processo, a melhor solução encontrada ao longo das iterações é armazenada como solução final do algoritmo (FEO; RESENDE, 1995).

Segundo Talbi (2009), o desempenho do *GRASP* depende diretamente do equilíbrio entre aleatoriedade e gulosidade durante a construção das soluções. Um nível reduzido de aleatoriedade pode limitar a exploração do espaço de busca, enquanto excesso de aleatoriedade pode comprometer a qualidade das soluções geradas.

O procedimento geral do *GRASP* é apresentado na FIG. 5.

Figura 5: Pseudocódigo do GRASP

```

procedimento GRASP( $f(\cdot), g(\cdot), N(\cdot), GRASP_{max}, s$ )
1   $f^* \leftarrow \infty$ ;
2  para ( $Iter = 1, 2, \dots, GRASP_{max}$ ) faça
3      Construcao( $g(\cdot), \alpha, s$ );
4      BuscaLocal( $f(\cdot), N(\cdot), s$ );
5      se ( $f(s) < f^*$ ) então
6           $s^* \leftarrow s$ ;
7           $f^* \leftarrow f(s)$ ;
8      fim-se;
9  fim-para;
10  $s \leftarrow s^*$ ;
11 Retorne  $s$ ;
fim GRASP

```

Fonte: Souza (2024, p. 32).

No pseudocódigo apresentado, $f(\cdot)$ representa a função objetivo, $g(\cdot)$ corresponde à função gulosa utilizada na construção da solução, $N(\cdot)$ define as estruturas de vizinhança, e $GRASP_{max}$ estabelece o número máximo de iterações. Na linha 1, o valor da melhor solução, representado por f^* , é inicialmente definido como infinito. A linha 2 inicia o laço de repetição, executado de 1 até $GRASP_{max}$. Na linha 3, uma solução s é gerada pelo procedimento de construção, utilizando a função gulosa $g(\cdot)$ e o parâmetro α , responsável por controlar o nível de gulosidade e aleatoriedade. Na linha 4, a solução construída é aprimorada pelo procedimento de busca local, considerando a função objetivo e as estruturas de vizinhança definidas.

Na linha 5, o *makespan* da solução obtida é comparado com o melhor valor encontrado até aquele momento. Caso $f(s) < f^*$, a linha 6 armazena s como a nova melhor solução s^* , enquanto a linha 7 atualiza o melhor valor da função objetivo. A linha 8 encerra essa condição e a linha 9 finaliza a iteração. Após a realização de todas as iterações, a linha 10 atribui a melhor solução encontrada a s , e a linha 11 retorna essa solução como resultado final do algoritmo (SOUZA, 2024, p. 32).

De modo geral, o *GRASP* destaca-se pela simplicidade de implementação e pela capacidade de gerar soluções diversificadas por meio da combinação entre critérios gulosos e aleatórios. Em problemas de sequenciamento, essa característica permite explorar diferentes regiões do espaço de soluções, contribuindo para a obtenção de soluções de boa qualidade em tempos computacionais reduzidos (RESENDE; RIBEIRO, 2016).

3.3.2 *Iterated Local Search (ILS)*

O *Iterated Local Search (ILS)* é uma meta-heurística baseada na aplicação sucessiva de procedimentos de busca local, proposta com o objetivo de explorar diferentes regiões do espaço de soluções de forma eficiente. Segundo Lourenço, Martin e Stützle (2003), o método parte da ideia de que soluções de boa qualidade podem ser obtidas por busca local, porém frequentemente ficam restritas a ótimos locais. Dessa forma, o *ILS* utiliza mecanismos de perturbação para escapar dessas regiões e continuar a exploração do espaço de busca.

O funcionamento do *ILS* inicia-se a partir da geração de uma solução inicial, sobre a qual é aplicada uma busca local até a obtenção de um ótimo local. Em seguida, a solução é modificada por meio de uma perturbação, gerando uma nova solução candidata. Essa nova solução passa novamente por uma etapa de busca local, repetindo o processo iterativamente (LOURENÇO; MARTIN; STÜTZLE, 2003).

Segundo Talbi (2009), a principal característica do *ILS* consiste na alternância entre intensificação e diversificação. A busca local atua na intensificação da busca em regiões promissoras, enquanto a perturbação permite explorar novas regiões do espaço de soluções, reduzindo a chance de aprisionamento em ótimos locais.

A eficiência do método está diretamente relacionada à definição da estratégia de perturbação utilizada. Perturbações muito pequenas podem fazer com que o algoritmo retorne ao mesmo ótimo local, enquanto perturbações excessivamente grandes podem descaracterizar completamente a solução obtida anteriormente, reduzindo a eficiência da busca (LOURENÇO; MARTIN; STÜTZLE, 2003).

De forma geral, o funcionamento do *ILS* consiste na repetição sucessiva das etapas de construção da solução inicial, busca local, perturbação e critério de aceitação. Ao final do processo iterativo, a melhor solução encontrada ao longo da execução é armazenada como solução final do algoritmo (LOURENÇO; MARTIN; STÜTZLE, 2003).

O funcionamento do *ILS* pode ser observado no pseudocódigo apresentado na FIG. 6.

Figura 6: Pseudocódigo do *ILS*

```

procedimento ILS
1   $s_0 \leftarrow \text{GeraSolucaoInicial}();$ 
2   $s \leftarrow \text{BuscaLocal}(s_0);$ 
3  enquanto (os critérios de parada não estiverem satisfeitos) faça
4       $s' \leftarrow \text{Perturbacao}(\text{histórico}, s);$ 
5       $s'' \leftarrow \text{BuscaLocal}(s');$ 
6       $s \leftarrow \text{CritérioAceitacao}(s, s'', \text{histórico});$ 
8  fim-enquanto;
fim ILS;

```

Fonte: Souza (2024, p. 36).

Na linha 1 do pseudocódigo, o procedimento *GeraSolucaoInicial* produz a solução inicial s_0 . Na linha 2, essa solução é submetida à busca local, resultando na solução corrente s . A linha 3 inicia o processo iterativo, que permanece em execução enquanto o critério de parada não for satisfeito. Esse critério pode ser definido, por exemplo, por um número máximo de iterações ou por um limite de tempo computacional.

Na linha 4, uma perturbação é aplicada à solução corrente, produzindo a solução intermediária s' . O histórico representa informações acumuladas durante a busca e pode auxiliar na definição da perturbação. Na linha 5, uma nova busca local é aplicada sobre s' , resultando na solução refinada s'' . Na linha 6, o critério de aceitação determina se a busca continuará a partir da solução corrente s ou da nova solução s'' . Por fim, a linha 8 encerra o laço de repetição, finalizando o algoritmo quando o critério de parada é alcançado (SOUZA, 2024, p. 36).

O *ILS* apresenta como principal vantagem a utilização de mecanismos simples de perturbação aliados à busca local. Essa combinação permite intensificar a exploração de soluções promissoras ao mesmo tempo em que reduz a probabilidade de aprisionamento em ótimos locais, tornando o método eficiente para problemas de sequenciamento de elevada complexidade (LOURENÇO; MARTIN; STÜTZLE, 2003).

3.3.3 *Variable Neighborhood Search (VNS)*

O método *Variable Neighborhood Search (VNS)* é uma meta-heurística proposta por Mladenović e Hansen (1997), baseada na exploração sistemática de diferentes estruturas de vizinhança durante o processo de busca. O método parte do princípio de que soluções ótimas locais em uma determinada vizinhança não necessariamente permanecem ótimas em outras estruturas de vizinhança.

Segundo Mladenović e Hansen (1997), o *VNS* utiliza múltiplas estruturas de vizinhança com o objetivo de ampliar a exploração do espaço de soluções e reduzir a chance de aprisionamento em ótimos locais. Dessa forma, o algoritmo alterna entre diferentes movimentos de vizinhança ao longo da execução, permitindo explorar diferentes regiões do espaço de busca.

O funcionamento do *VNS* inicia-se a partir de uma solução inicial. Em seguida, é aplicada uma perturbação na solução corrente por meio de uma estrutura de vizinhança previamente definida, etapa conhecida como *shaking*. Após essa perturbação, aplica-se uma busca local com o objetivo de refinar a solução obtida (MLADENOVIĆ; HANSEN, 1997).

Caso a nova solução apresente melhora em relação à solução atual, ela passa a ser a nova solução corrente e o algoritmo retorna à primeira estrutura de vizinhança. Caso contrário, o método avança para a próxima vizinhança disponível, ampliando gradualmente a exploração do espaço de soluções (TALBI, 2009).

Segundo Talbi (2009), a principal característica do *VNS* está na combinação entre intensificação e diversificação da busca. Enquanto a busca local intensifica a exploração em regiões promissoras, a mudança sistemática das estruturas de vizinhança favorece a diversificação, permitindo explorar diferentes regiões do espaço de soluções.

De forma geral, o *VNS* consiste na repetição sucessiva das etapas de perturbação, busca local e mudança de vizinhança, armazenando ao final da execução a melhor solução encontrada durante o processo iterativo (MLADENOVIĆ; HANSEN, 1997).

O pseudocódigo geral do *VNS* é apresentado na FIG. 7.

Figura 7: Pseudocódigo do VNS

```

procedimento VNS()
1  Seja  $s_0$  uma solução inicial;
2  Seja  $r$  o número de estruturas diferentes de vizinhança;
3   $s \leftarrow s_0$ ;           {Solução corrente}
4  enquanto (Critério de parada não for satisfeito) faça
5       $k \leftarrow 1$ ;       {Tipo de estrutura de vizinhança corrente}
6      enquanto ( $k \leq r$ ) faça
7          Gere um vizinho qualquer  $s' \in N^{(k)}(s)$ ;
8           $s'' \leftarrow \text{BuscaLocal}(s')$ ;
9          se ( $f(s'') < f(s)$ )
10             então
11                  $s \leftarrow s''$ ;
12                  $k \leftarrow 1$ ;
13             senão
14                  $k \leftarrow k + 1$ ;
15         fim-se;
16     fim-enquanto;
17 fim-enquanto;
18 Retorne  $s$ ;
fim VNS;

```

Fonte: Souza (2024, p. 35).

No pseudocódigo apresentado na FIG. 7, na linha 1, s_0 é definida como a solução inicial. Na linha 2, r recebe o número de estruturas de vizinhança disponíveis. Na linha 3, a solução inicial é armazenada como solução corrente s . A linha 4 inicia o laço principal, executado enquanto o critério de parada não for satisfeito. Na linha 5, o índice k é definido como 1, indicando a primeira estrutura de vizinhança. A linha 6 inicia a exploração das vizinhanças, que continua enquanto $k \leq r$.

Na linha 7, uma solução vizinha s' é gerada a partir da k -ésima estrutura de vizinhança da solução corrente. Na linha 8, a busca local é aplicada sobre s' , produzindo a solução refinada s'' . Na linha 9, verifica-se se s'' apresenta valor da função objetivo menor que o da solução corrente. Caso haja melhoria, as linhas 10 e 11 armazenam s'' como a nova solução corrente, e a linha 12 reinicia a exploração a partir da primeira vizinhança. Caso contrário, as linhas 13 e 14 incrementam k , conduzindo a busca para a próxima estrutura de vizinhança.

A linha 15 encerra a condição de melhoria, a linha 16 finaliza a exploração das vizinhanças e a linha 17 encerra o laço principal. Finalmente, a linha 18 retorna a melhor solução corrente encontrada pelo algoritmo (SOUZA, 2024, p. 35).

A principal característica do *VNS* consiste na utilização sistemática de múltiplas estruturas de vizinhança, permitindo alternar entre diferentes mecanismos de exploração do espaço de busca. Essa estratégia favorece o equilíbrio entre intensificação e diversificação, contribuindo para a obtenção de soluções de elevada qualidade em problemas combinatórios complexos (TALBI, 2009).

3.3.4 *Simulated Annealing (SA)*

O *Simulated Annealing (SA)*, ou Recozimento Simulado, é uma meta-heurística de busca local probabilística proposta por Kirkpatrick, Gelatt e Vecchi (1983), inspirada no processo de recozimento térmico utilizado na metalurgia. Segundo Lopes, Rodrigues e Steiner (2013), nesse processo o material é aquecido a altas temperaturas e resfriado lentamente, permitindo que os átomos atinjam estados de menor energia e maior estabilidade. Na otimização, essa analogia relaciona a energia do sistema ao valor da função objetivo, enquanto o resfriamento representa a busca gradual por soluções de melhor qualidade.

A principal característica do *Simulated Annealing* é permitir, de forma probabilística, a aceitação de soluções de pior qualidade ao longo da busca (KIRKPATRICK; GELATT; VECCHI, 1983). Diferentemente dos métodos clássicos de busca local, que aceitam apenas soluções de melhoria, o *SA* permite movimentos para soluções inferiores, reduzindo a chance de aprisionamento em ótimos locais.

Segundo Lopes, Rodrigues e Steiner (2013), o *Simulated Annealing* pode ser entendido como uma extensão dos métodos de busca local. O algoritmo explora soluções vizinhas por meio de pequenas modificações na solução atual, utilizando estruturas de vizinhança previamente definidas. Dessa forma, o método combina diversificação e intensificação ao longo da execução.

O método inicia-se a partir de uma solução inicial e de uma temperatura inicial elevada. A cada iteração, uma nova solução vizinha é gerada por meio de pequenas modificações na solução corrente. Caso a solução gerada apresente melhora na função objetivo, ela é automaticamente aceita. Entretanto, quando ocorre piora na solução, sua aceitação depende de um critério

probabilístico associado à variação da função objetivo e à temperatura atual do sistema (KIRKPATRICK; GELATT; VECCHI, 1983).

Segundo Černý (1985), a aceitação de soluções inferiores ocorre com uma probabilidade exponencialmente decrescente. Assim, temperaturas elevadas favorecem maior exploração do espaço de busca, enquanto temperaturas menores tornam o algoritmo mais restritivo, intensificando a busca em regiões promissoras.

A probabilidade de aceitação de soluções inferiores é calculada pela Equação de Boltzmann, apresentada na Equação 1:

$$P = e^{-\Delta/T} \quad (1)$$

Em que Δ representa a variação da função objetivo entre a solução atual e a solução vizinha, enquanto T corresponde à temperatura corrente do sistema. Dessa forma, quanto maior a temperatura, maior será a chance de aceitação de soluções inferiores.

Para decidir se uma solução inferior será aceita, gera-se um número aleatório r a partir de uma distribuição uniforme no intervalo $[0,1]$. Caso r seja menor ou igual a P , a nova solução é aceita; caso contrário, a solução é rejeitada. Assim, quanto maior o valor de P , maior será a chance de o algoritmo aceitar movimentos de piora (LOPES; RODRIGUES; STEINER, 2013).

Durante a execução do algoritmo, a temperatura é gradativamente reduzida por meio de uma taxa de resfriamento α , em que $0 < \alpha < 1$. Valores baixos de α promovem resfriamento mais rápido, reduzindo o tempo computacional, porém podendo comprometer a qualidade das soluções obtidas. Por outro lado, valores de α próximos de 1 tornam o resfriamento mais lento, favorecendo uma exploração mais ampla do espaço de busca e aumentando a possibilidade de obtenção de melhores soluções (LOPES; RODRIGUES; STEINER, 2013).

Além disso, a cada redução de temperatura, o algoritmo executa um número pré-definido de iterações visando restabelecer o equilíbrio térmico do sistema. O procedimento é encerrado quando a temperatura atinge valores próximos de zero ou quando não ocorrem melhorias significativas na solução ao longo das iterações (TALBI, 2009).

Segundo Talbi (2009), o desempenho do *Simulated Annealing* está diretamente relacionado à definição adequada de seus parâmetros. Entre os principais parâmetros do método destacam-se a temperatura inicial, a taxa de resfriamento, o número de iterações executadas em cada

temperatura e o critério de parada. A escolha inadequada desses parâmetros pode comprometer tanto a qualidade das soluções quanto o desempenho computacional do algoritmo.

Além disso, Lopes, Rodrigues e Steiner (2013) destacam que o equilíbrio entre exploração e intensificação é um dos principais aspectos das meta-heurísticas modernas. No *Simulated Annealing*, esse equilíbrio é controlado principalmente pelo mecanismo de resfriamento, responsável por reduzir gradualmente a aceitação de soluções de piora ao longo da execução do algoritmo.

O algoritmo do *Simulated Annealing* é apresentado na FIG. 8.

Figura 8: Pseudocódigo do SA

```

procedimento  $SA(f(.), N(.), \alpha, SAmax, T_0, s)$ 
1   $s^* \leftarrow s;$            {Melhor solução obtida até então}
2   $IterT \leftarrow 0;$        {Número de iterações na temperatura T}
3   $T \leftarrow T_0;$          {Temperatura corrente}
4  enquanto  $(T > 0)$  faça
5      enquanto  $(IterT < SAmax)$  faça
6           $IterT \leftarrow IterT + 1;$ 
7          Gere um vizinho qualquer  $s' \in N(s);$ 
8           $\Delta = f(s') - f(s);$ 
9          se  $(\Delta < 0)$ 
10             então
11                  $s \leftarrow s';$ 
12                 se  $(f(s') < f(s^*))$  então  $s^* \leftarrow s';$ 
13             senão
14                 Tome  $x \in [0, 1];$ 
15                 se  $(x < e^{-\Delta/T})$  então  $s \leftarrow s';$ 
16             fim-se;
17         fim-enquanto;
18          $T \leftarrow \alpha \times T;$ 
19          $IterT \leftarrow 0;$ 
20     fim-enquanto;
21      $s \leftarrow s^*;$ 
22     Retorne  $s;$ 
fim  $SA;$ 

```

No pseudocódigo apresentado na FIG. 8, $f(\cdot)$ representa a função objetivo, $N(\cdot)$ define as estruturas de vizinhança, α corresponde à taxa de resfriamento, $SAmax$ estabelece o número de iterações em cada temperatura, e T_0 representa a temperatura inicial. Na linha 1, a solução inicial s é armazenada como a melhor solução encontrada s^* . Na linha 2, o contador de iterações na temperatura corrente é iniciado em zero. Na linha 3, a temperatura (T) recebe o valor da temperatura inicial T_0 . A linha 4 inicia o laço de resfriamento, enquanto a linha 5 controla a realização de $SAmax$ iterações em cada temperatura.

Na linha 6, o contador $IterT$ é incrementado. Na linha 7, uma solução vizinha s' é gerada a partir da solução corrente s . Na linha 8, calcula-se $\Delta = f(s') - f(s)$, que representa a variação do *makespan* provocada pelo movimento. A linha 9 verifica se $\Delta < 0$, indicando que a solução vizinha possui menor *makespan*. Nesse caso, as linhas 10 e 11 aceitam s' como a nova solução corrente. Na linha 12, verifica-se se essa solução também é melhor que s^* ; caso seja, ela passa a ser armazenada como a melhor solução encontrada.

Quando $\Delta \geq 0$, a linha 13 conduz o algoritmo ao critério probabilístico de aceitação. Na linha 14, é gerado um número aleatório x no intervalo $[0, 1]$. Na linha 15, a solução vizinha é aceita quando $x < e^{-\Delta/T}$. Esse mecanismo permite aceitar movimentos de piora, principalmente nas temperaturas mais elevadas. A linha 16 encerra a condição de aceitação, e a linha 17 finaliza as iterações realizadas na temperatura corrente.

Na linha 18, a temperatura é reduzida por meio da expressão $T = \alpha T$, e, na linha 19, o contador $IterT$ é reiniciado. A linha 20 encerra o processo de resfriamento. Por fim, a linha 21 atribui a melhor solução encontrada a s , e a linha 22 retorna essa solução como resultado do algoritmo. Na implementação deste trabalho, o procedimento é encerrado quando $T \leq T_f$.

Entre as principais vantagens do *Simulated Annealing* destaca-se sua capacidade de escapar de ótimos locais por meio da aceitação probabilística de soluções inferiores durante as fases iniciais da busca. Essa característica torna o método particularmente adequado para problemas de sequenciamento com elevado número de soluções viáveis e grande complexidade combinatória (LOPES; RODRIGUES; STEINER, 2013).

4 ESTUDOS E APLICAÇÕES DO PROBLEMA DE MÁQUINAS PARALELAS IDÊNTICAS COM RESTRIÇÃO DE RECURSOS

O problema de máquinas paralelas idênticas com restrição de recursos envolve duas decisões principais: a distribuição das tarefas entre as máquinas disponíveis e a definição da ordem em que elas serão processadas. Além da disponibilidade de uma máquina, cada tarefa necessita de um recurso adicional durante sua execução. Como esses recursos possuem disponibilidade limitada, tarefas que compartilham o mesmo recurso não podem ser processadas simultaneamente, mesmo quando estão alocadas em máquinas diferentes (CHUNG *et al.*, 2019).

Diversos estudos têm proposto modelos matemáticos, heurísticas e meta-heurísticas para resolver diferentes variantes do problema, considerando restrições relacionadas ao compartilhamento de recursos e buscando obter soluções de elevada qualidade em tempos computacionais viáveis.

Um dos trabalhos clássicos na área foi desenvolvido por Ventura e Kim (2003), que estudaram o problema de sequenciamento em máquinas paralelas idênticas com recursos adicionais limitados e penalidades de adiantamento e atraso. Os autores formularam o problema por meio de programação linear inteira binária e propuseram uma abordagem baseada em relaxação Lagrangeana para obter soluções de boa qualidade e avaliar seu desempenho. Os experimentos foram realizados em instâncias com até 300 tarefas e demonstraram a eficácia da abordagem proposta, que obteve desvio médio de apenas 0,63% em relação aos limitantes inferiores e encontrou soluções ótimas em 174 das 270 instâncias avaliadas. Os resultados evidenciaram a complexidade introduzida pelos recursos compartilhados e o potencial de métodos híbridos para a resolução de problemas de maior porte.

Posteriormente, Chung *et al.* (2019) estudaram a minimização do *makespan* em duas máquinas paralelas idênticas com restrições de moldes (recursos), motivados por uma aplicação na indústria de semicondutores. Os autores propuseram um modelo de Programação Linear Inteira Mista e desenvolveram as heurísticas *TLPT* e *DMLPT*, ambas baseadas na regra *LPT*, apresentada na Seção 3.2 deste trabalho. Além disso, propuseram a heurística combinada *CTD*, que executa as duas abordagens e seleciona a solução com o menor *makespan*.

Para avaliar os métodos propostos, os autores realizaram comparações com as meta-heurísticas *VNS* e *DPSO* (*Discrete Particle Swarm Optimization*), uma adaptação discreta da otimização por enxame de partículas. Os experimentos, realizados em 580 instâncias, mostraram que a

CTD apresentou desvio percentual médio próximo de 0,20% nas instâncias de maior porte e utilizou menos de 5% do tempo computacional requerido pela *VNS* e pela *DPSO* (CHUNG *et al.*, 2019).

No contexto de sistemas flexíveis de manufatura, Soares e Carvalho (2020) investigaram o problema de máquinas paralelas idênticas com restrições de ferramentas, uma variante relacionada ao problema de máquinas paralelas com recursos compartilhados. Para a resolução do problema, os autores propuseram um algoritmo genético de chaves aleatórias enviesadas (*Biased Random-Key Genetic Algorithm - BRKGA*) híbrido, combinado com procedimentos de busca local organizados por *Variable Neighborhood Descent (VND)*. Os experimentos computacionais foram realizados em 2.880 instâncias de *benchmark* e demonstraram que o método proposto superou o estado da arte da literatura, produzindo soluções melhores ou equivalentes em 99,86% das instâncias de pequeno porte e novas melhores soluções para 93,89% das instâncias de grande porte. Além disso, o algoritmo apresentou redução média de 96,5% no tempo computacional em comparação ao método de referência, evidenciando sua eficiência e robustez na resolução do problema (SOARES; CARVALHO, 2020).

Dando continuidade a essa linha de pesquisa, Soares e Carvalho (2022) estudaram o problema de máquinas paralelas idênticas com recursos compartilhados e tempos de preparação dependentes da sequência, motivado por aplicações na fabricação de componentes microeletrônicos. Para resolver o problema, os autores propuseram uma meta-heurística híbrida baseada em *BRKGA* e *VND*. Os resultados demonstraram a eficácia da abordagem, que obteve os resultados ótimos para todas as instâncias de *benchmark* da literatura e apresentou desempenho competitivo em um novo conjunto de 270 instâncias de maior porte e complexidade propostos pelos autores (SOARES; CARVALHO, 2022).

Mais recentemente, Zhang *et al.* (2023) estudaram uma variante do problema de máquinas paralelas com recursos compartilhados, na qual cada tarefa necessita simultaneamente de uma máquina e de um recurso adicional durante todo o seu processamento. Os autores desenvolveram um método de solução com garantias teóricas de desempenho e realizaram experimentos computacionais para avaliar sua eficiência. Os resultados demonstraram que a abordagem proposta foi capaz de obter soluções de alta qualidade para diferentes configurações do problema, contribuindo para o avanço dos métodos de resolução de problemas de sequenciamento com restrições de recursos (ZHANG *et al.*, 2023).

De maneira geral, observa-se uma evolução da literatura desde abordagens baseadas em modelos matemáticos exatos para métodos heurísticos e meta-heurísticos cada vez mais sofisticados. Enquanto os métodos exatos apresentam desempenho adequado para instâncias de pequeno porte, as meta-heurísticas têm demonstrado maior capacidade de resolver problemas reais de grande dimensão, produzindo soluções de elevada qualidade em tempos computacionais reduzidos. Nesse contexto, destacam-se técnicas como Algoritmos Genéticos, Busca Tabu, *Simulated Annealing*, *Variable Neighborhood Search* e suas versões híbridas com mecanismos de busca local (CHUNG *et al.*, 2019; SOARES; CARVALHO, 2020; SOARES; CARVALHO, 2022).

Apesar dos avanços observados, ainda são escassos os estudos que realizam comparações sistemáticas entre diferentes meta-heurísticas clássicas aplicadas a um mesmo conjunto de instâncias do PMPIRR. Além disso, poucos trabalhos investigam simultaneamente o impacto da calibração de parâmetros e do porte das instâncias sobre o desempenho dos algoritmos. Dessa forma, este trabalho busca contribuir para a literatura por meio da implementação e comparação das meta-heurísticas *GRASP*, *ILS*, *VNS* e *Simulated Annealing*, avaliando seu desempenho em diferentes cenários e tamanhos de instância.

5 METODOLOGIA

Este capítulo apresenta os procedimentos metodológicos adotados para o desenvolvimento desta pesquisa. Inicialmente, é realizada a caracterização do problema estudado, bem como a descrição do ambiente computacional, da representação das soluções, das estruturas de vizinhança e da heurística utilizada para a geração das soluções iniciais. Em seguida, são apresentados os algoritmos meta-heurísticos implementados e o processo de calibração de seus parâmetros por meio de planejamento de experimentos.

Após a etapa de calibração, as meta-heurísticas são avaliadas utilizando instâncias clássicas da literatura e instâncias geradas especificamente para este trabalho, contemplando diferentes configurações do problema quanto ao número de tarefas, máquinas e recursos compartilhados. Dessa forma, torna-se possível analisar o comportamento dos métodos em cenários com diferentes níveis de complexidade.

Por fim, os resultados obtidos são comparados entre si e com um modelo de Programação Linear Inteira Mista resolvido pelo *Gurobi*, considerando a qualidade das soluções encontradas e o desempenho computacional de cada abordagem. As etapas da metodologia são apresentadas nas seções seguintes.

5.1 Caracterização do problema

O problema estudado neste trabalho corresponde ao problema de máquinas paralelas idênticas com restrição de recursos, no qual um conjunto de tarefas deve ser distribuído entre máquinas paralelas disponíveis para processamento.

Além das restrições associadas às máquinas, cada tarefa necessita da utilização de um recurso adicional durante sua execução. Como esses recursos possuem disponibilidade limitada, tarefas que utilizam o mesmo recurso não podem ser processadas simultaneamente, mesmo que existam máquinas livres no sistema.

Neste trabalho, considera-se que todas as tarefas estão disponíveis para processamento no instante inicial, cada tarefa possui apenas uma operação, cada máquina pode processar apenas uma tarefa por vez e não é permitida a interrupção do processamento das tarefas (não preempção). Assim, uma vez iniciada, a execução de uma tarefa deve prosseguir até sua conclusão.

O objetivo consiste em determinar a alocação e o sequenciamento das tarefas de forma a minimizar o *makespan* (C_{max}) e representa o instante de conclusão da última tarefa do cronograma.

Devido à elevada complexidade desse problema, especialmente para instâncias de maior porte, foram utilizadas meta-heurísticas como alternativa para obtenção de soluções de boa qualidade em tempos computacionais viáveis.

Os experimentos foram realizados em um computador equipado com processador Intel® Core™ i5-9400F CPU @ 2,90 GHz, 16 GB de memória RAM e sistema operacional *Windows 11 Home Single Language* 64 bits.

5.2 Função de avaliação

A função de avaliação foi utilizada para comparar as soluções geradas pelas meta-heurísticas. O valor de cada solução foi calculado pela soma do *makespan* com uma penalidade associada aos conflitos no uso dos recursos compartilhados, conforme apresentado na Equação 2:

$$f(s) = C_{(max)}(s) + \lambda \sum_{r \in R} confr_r(s) \quad (2)$$

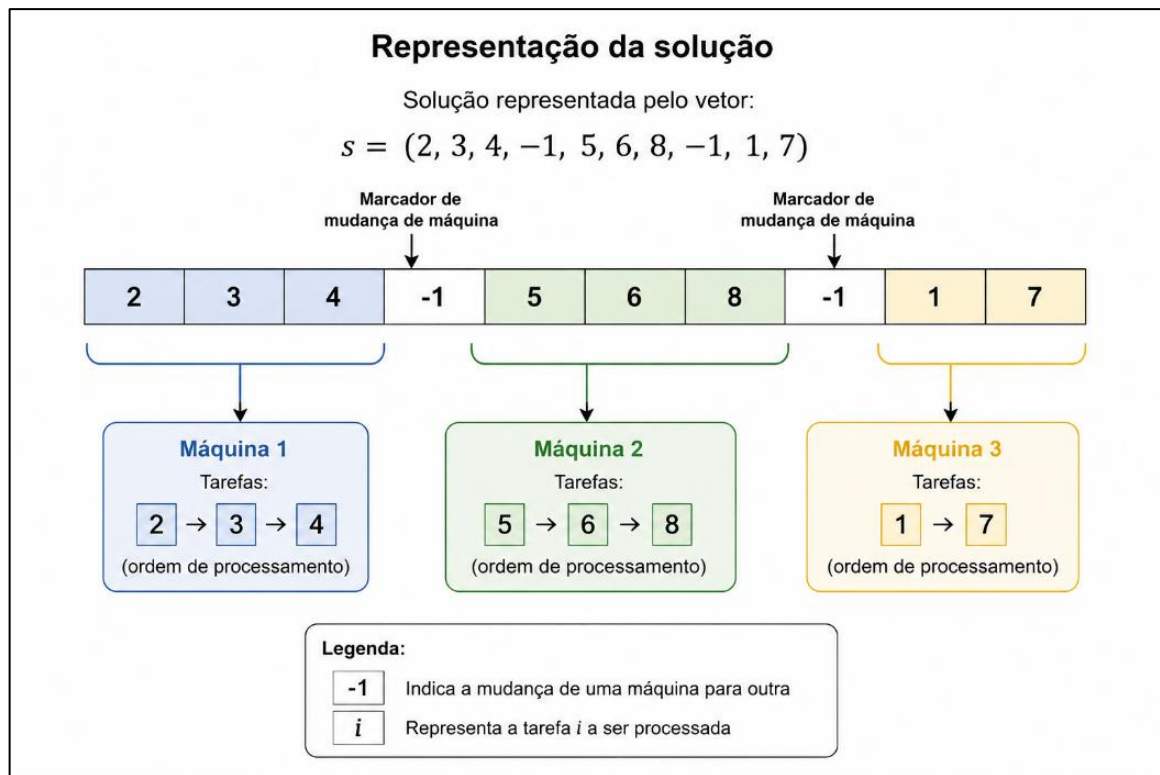
em que $confr_r(s)$ corresponde à quantidade de conflitos envolvendo o recurso r , e λ representa o peso da penalidade. Quando não há conflitos, a avaliação corresponde apenas ao *makespan*. Caso contrário, a penalidade aumenta o valor da função, desfavorecendo a solução. Como o problema é de minimização, quanto menor o valor de $f(s)$, melhor é a solução.

5.3 Representação da solução

Para a aplicação das meta-heurísticas, foi necessário representar simultaneamente a alocação das tarefas às máquinas e a sequência de processamento adotada em cada uma delas. Dessa forma, cada solução foi representada por um vetor contendo todas as tarefas do problema, utilizando marcadores para indicar a separação entre as máquinas.

A FIG. 9 apresenta um exemplo dessa representação para uma instância composta por oito tarefas distribuídas em três máquinas. Nessa estrutura, o valor "-1" é utilizado para indicar a mudança de uma máquina para outra. Assim, as tarefas localizadas antes do primeiro marcador são processadas pela primeira máquina, as tarefas localizadas entre o primeiro e o segundo marcador são processadas pela segunda máquina e as tarefas após o segundo marcador são processadas pela terceira máquina.

Figura 9: Representação da solução utilizada pelas meta-heurísticas



Fonte: Elaboração própria com auxílio de inteligência artificial (2026).

5.4 Estruturas de vizinhanças

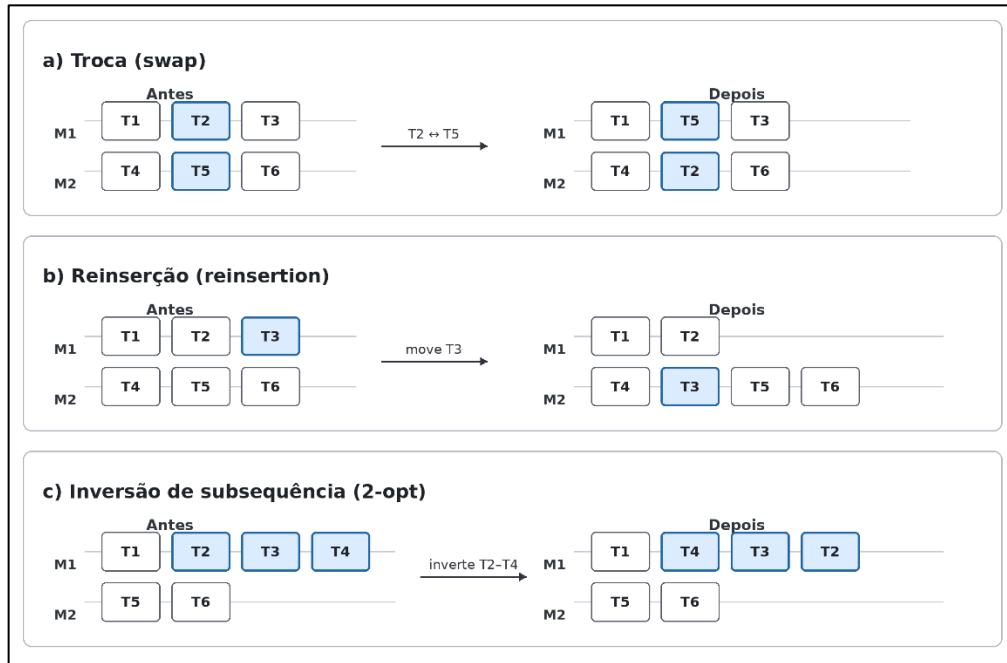
As estruturas de vizinhança são responsáveis por gerar novas soluções a partir de uma solução já existente. Em problemas de sequenciamento, esses movimentos permitem modificar a alocação das tarefas entre as máquinas e a ordem em que elas são processadas, possibilitando que as meta-heurísticas explorem diferentes regiões do espaço de busca.

Neste trabalho, foram utilizadas três estruturas de vizinhança, cujos conceitos foram apresentados na revisão bibliográfica: troca (*swap*), reinserção (*reinsertion*) e inversão de subsequência (*2-opt*).

No movimento de troca, duas tarefas são selecionadas e trocadas entre si, podendo modificar tanto a ordem de processamento quanto a máquina à qual estão atribuídas. Na reinserção, uma tarefa é removida de sua posição original e inserida em outra posição da solução. Por sua vez, no movimento 2-opt, são selecionadas duas posições e a ordem das tarefas pertencentes à subsequência delimitada por elas é invertida.

A FIG. 10 apresenta exemplos simplificados da aplicação dessas três estruturas de vizinhança em uma solução composta por duas máquinas.

Figura 10: Exemplos das estruturas de vizinhança utilizadas



Fonte: Elaboração própria com auxílio de inteligência artificial (2026).

No exemplo de troca (a), as tarefas T_2 e T_5 têm suas posições permutadas entre as máquinas. Na reinserção (b), a tarefa T_3 é retirada da máquina 1 e inserida entre as tarefas T_4 e T_5 da máquina 2. Por fim, no movimento 2-opt (c), a subsequência formada por T_2 , T_3 e T_4 é invertida, passando a apresentar a ordem T_4 , T_3 e T_2 .

5.5 Solução inicial

As soluções iniciais utilizadas pelas meta-heurísticas foram geradas por meio de uma heurística construtiva baseada no método *LPT*, apresentado na Seção 3.2.

A cada iteração, são analisadas as tarefas ainda não alocadas e é formada uma Lista Restrita de Candidatos (LRC), controlada pelo parâmetro α . Em seguida, uma tarefa é selecionada aleatoriamente da LRC e alocada na máquina com menor carga acumulada no momento.

Esse processo é repetido até que todas as tarefas sejam distribuídas entre as máquinas. Ao combinar o critério guloso do *LPT* com uma escolha aleatória controlada, a estratégia permite gerar soluções iniciais diversificadas, contribuindo para a exploração do espaço de busca pelas meta-heurísticas implementadas.

5.6 Implementação dos algoritmos

Foram implementadas em *Python* as meta-heurísticas *GRASP*, *ILS*, *VNS* e *Simulated Annealing* (*SA*). Os pseudocódigos apresentados no Capítulo 3 representam as estruturas gerais desses métodos, enquanto esta seção descreve, de forma resumida, as versões efetivamente utilizadas nos experimentos.

Quando empregada, a busca local *VND* explora as vizinhanças na ordem 2-opt, troca e reinserção, retornando à primeira estrutura sempre que uma melhoria é encontrada.

O *GRASP* gera, a cada iteração, uma nova solução pelo procedimento *LPT* guloso-randomizado, posteriormente aprimorada por uma busca local *VND* composta pelas estruturas 2-opt, troca e reinserção. Sempre que uma solução melhor é encontrada, ela é armazenada como a melhor solução e o contador de iterações sem melhoria é reiniciado. A execução termina quando o critério máximo de iterações consecutivas sem melhoria ou o limite de tempo é atingido.

Para o *ILS*, a solução inicial também é construída pelo procedimento *LPT* guloso-randomizado e refinada pelo *VND*. A diversificação ocorre por meio de perturbações formadas por trocas aleatórias entre posições da solução, cuja intensidade aumenta após períodos consecutivos sem melhoria. Depois de cada perturbação, o *VND* é novamente aplicado, e a solução é atualizada quando apresenta melhoria. O critério de parada considera o número máximo de iterações consecutivas sem melhoria ou o limite de tempo.

A implementação do *VNS* utiliza as estruturas de troca, reinserção e 2-opt, cuja ordem de exploração é embaralhada aleatoriamente a cada ciclo. Cada solução gerada é submetida ao *VND* e, caso apresente melhoria, passa a ser a solução corrente e pode atualizar a melhor solução encontrada. Na ausência de melhoria, a próxima estrutura da lista é explorada. A execução prossegue até que o critério de parada por tentativas consecutivas sem melhoria ou por tempo seja alcançado.

Por fim, o *Simulated Annealing* parte de uma solução construída pelo *LPT* e gera soluções vizinhas por meio de movimentos aleatórios de troca ou reinserção. Soluções melhores são aceitas diretamente, enquanto soluções piores podem ser aceitas segundo a probabilidade de Boltzmann. A temperatura inicial é determinada de forma adaptativa, buscando uma taxa de aceitação de pelo menos 95%, e posteriormente reduzida até a temperatura final estabelecida. Ao término do processo, a melhor solução encontrada é refinada pelo *VND*.

5.7 Parâmetros a serem avaliados

A calibração dos parâmetros dos algoritmos não foi realizada de maneira arbitrária, mas fundamentada nos princípios de Planejamento e Análise de Experimentos, conforme discutido por Montgomery e Runger (2003). Essa abordagem permitiu identificar quais variáveis controláveis (parâmetros) possuem maior influência no desempenho das soluções para o problema.

Para encontrar a combinação ideal dos parâmetros, foi adotado o modelo de Experimento Fatorial. Esse método é considerado o mais eficiente por investigar o efeito conjunto de diferentes parâmetros ao testar todas as combinações possíveis entre seus níveis. Os testes foram organizados da seguinte forma:

- **GRASP e ILS:** Foram testadas todas as 30 combinações possíveis entre dois fatores principais:
 - **Número de iterações:** Diferentemente de uma abordagem com valores fixos (como 10 ou 20 iterações), optou-se por avaliar esse parâmetro de forma proporcional ao número de tarefas de cada instância (n_j). Foram estabelecidos cinco níveis: (i) $0,5 \times n_j$; (ii) $1 \times n_j$; (iii) $1,5 \times n_j$; (iv) $2 \times n_j$; e (v) $2,5 \times n_j$. Essa escolha fundamenta-se na dissertação de Corrêa (2025), que demonstra que o tipo de problema estudado torna-se significativamente mais complexo à medida que seu tamanho aumenta, sendo, portanto, necessárias mais iterações para a obtenção de soluções com melhor qualidade.
 - **Coefficiente α :** Testado em seis níveis: 0; 0,2; 0,4; 0,6; 0,8; e 1,0, visando equilibrar os componentes de gulosidade e aleatoriedade presentes nos métodos.
- **VNS:** O planejamento seguiu uma linha similar aos demais métodos, focando em dois fatores de influência:
 - **Número de iterações:** Assim como no *GRASP* e *ILS*, mantiveram-se os mesmos cinco níveis proporcionais ao tamanho da instância ($0,5 \times n_j$ a $2,5 \times n_j$), respeitando a premissa de que instâncias maiores exigem maior esforço computacional.
 - **Estrutura de vizinhanças:** Este fator foi avaliado por meio de uma estratégia de busca local dinâmica, na qual as vizinhanças foram organizadas em uma lista com ordem aleatória a cada ciclo. As estruturas foram então exploradas sequencialmente, sem repetição, até o esgotamento da lista. Caso uma solução

de melhoria fosse encontrada, o processo era reiniciado, permitindo intensificar a busca na região promissora do espaço de soluções.

- ***Simulated Annealing***: Foram avaliadas todas as 50 combinações possíveis entre os níveis dos parâmetros considerados no experimento, resultantes da combinação entre cinco níveis para o número de iterações por temperatura (SA_{max}) e dez níveis para a taxa de resfriamento (α). O planejamento experimental considerou os principais parâmetros responsáveis pelo controle do processo de resfriamento do algoritmo.
 - **Número de iterações por temperatura (SA_{max})**: Foram utilizados os mesmos níveis avaliados nos algoritmos *GRASP*, *ILS* e *VNS*.
 - **Taxa de resfriamento (α)**: Foram avaliados dez níveis para esse parâmetro, correspondentes aos valores 0,90; 0,91; 0,92; 0,93; 0,94; 0,95; 0,96; 0,97; 0,98 e 0,99, com o objetivo de analisar o impacto da velocidade de resfriamento no desempenho do algoritmo.
 - **Temperatura final (T_f)**: Foi fixada em 0,001 durante os experimentos de calibração.
 - **Temperatura inicial (T_0)**: Diferentemente dos demais parâmetros, a temperatura inicial não foi calibrada manualmente. Foi utilizado um procedimento adaptativo, baseado em Souza (2024), no qual a temperatura é inicialmente definida como $T_0 = 1$ e, a cada iteração, multiplicada por um fator $\beta = 2$ até que a taxa de aceitação de movimentos atinja pelo menos 95%. Dessa forma, a temperatura inicial é ajustada automaticamente de acordo com as características da instância analisada.

Dessa forma, a seleção dos parâmetros a serem testados foi conduzida de maneira sistemática, permitindo avaliar o impacto de cada fator no desempenho dos algoritmos. A partir dos resultados obtidos nos experimentos, tornou-se possível identificar as combinações de parâmetros mais adequadas para cada método, as quais serão abordadas na etapa de definição apresentada na seção seguinte.

5.7.1 Análise e definição dos parâmetros

Para a definição dos parâmetros, inicialmente foram geradas 10 instâncias de teste representando diferentes configurações do problema de sequenciamento em máquinas paralelas com restrição de recursos. As instâncias foram construídas variando o número de tarefas,

máquinas e recursos, de forma a contemplar diferentes níveis de complexidade do problema. A geração dessas instâncias foi realizada com o auxílio do *ChatGPT*, sendo utilizadas diretamente nos experimentos.

Foram considerados cenários com 20, 35, 40 e 50 tarefas, combinados com diferentes quantidades de máquinas (entre 2 e 5) e recursos (entre 3 e 6). A TAB. 1 apresenta a configuração das instâncias utilizadas neste estudo.

Tabela 1 - Instâncias de teste utilizadas na definição dos parâmetros

Instância	Nº de tarefas	Nº de máquinas	Nº de recursos
I1	20	2	3
I2	20	2	4
I3	20	3	4
I4	20	3	5
I5	35	3	4
I6	35	3	5
I7	40	4	5
I8	40	4	6
I9	50	4	5
I10	50	5	6

Fonte: Elaborada pelo autor (2026).

Com base nessas instâncias, os algoritmos foram executados combinando os parâmetros apresentados na etapa anterior. Para cada combinação de parâmetros foram realizadas 30 execuções, com o objetivo de analisar estatisticamente os resultados, reduzindo a influência de variações aleatórias nas execuções.

Para controlar a aleatoriedade das execuções, foi utilizada uma semente distinta em cada replicação. No código implementado, uma semente-base é definida a partir do instante de execução do programa e, para cada replicação, a semente utilizada é obtida pela soma entre essa semente-base e o número da replicação. Os valores das sementes são registrados automaticamente junto aos resultados de cada execução, permitindo a reprodução das respectivas soluções obtidas.

Os resultados de cada execução foram gerados automaticamente pelo código desenvolvido, sendo exportados em arquivos no formato *CSV*. Em seguida, esses dados foram exportados para planilhas eletrônicas (*Microsoft Excel*), nas quais foram realizadas análises estatísticas,

incluindo o cálculo da média, variância e desvio padrão do *makespan* e tempo de execução para cada combinação de parâmetros.

5.7.2 Análise e definição dos parâmetros do *GRASP*

A análise dos resultados obtidos pelo algoritmo *GRASP* revelou um comportamento distinto entre instâncias de pequeno porte e instâncias de maior complexidade. O desempenho foi avaliado sob a ótica do *makespan* e do tempo computacional, buscando identificar o ponto de equilíbrio ideal no Experimento Fatorial realizado.

Para as instâncias de pequeno porte (I1 a I4, com 20 tarefas), observou-se uma execução rápida. Em muitos casos, diferentes níveis de α e de iterações resultaram no mesmo *makespan*. Nessas situações o critério de desempate adotado foi o tempo computacional, privilegiando combinações que atingiam o menor *makespan* observado em menor tempo.

À medida que o tamanho das instâncias aumentou para 35, 40 e 50 tarefas (I5 a I10), o impacto da gulosidade controlada tornou-se mais evidente. Nesses cenários, os principais pontos observados foram:

- **Iterações:** Nas instâncias de 40 e 50 tarefas, observou-se que a configuração de $2,5 \times n_j$ foi essencial. Devido à maior amplitude do espaço de busca em problemas maiores, esse aumento no número de tentativas permitiu que o algoritmo explorasse melhor as combinações, resultando em menores *makespans* médios e menor variabilidade entre as execuções. Embora essa escolha resulte em um tempo de processamento superior às demais configurações, o ganho na qualidade e na confiabilidade das soluções justifica o custo computacional adicional para instâncias desse porte.
- **Aleatoriedade (Coeficiente α):** Enquanto instâncias com 35 tarefas (I5 e I6) apresentaram bom desempenho com $\alpha = 0,8$, as instâncias de grande porte (I9 e I10) demonstraram melhores resultados com valores de α entre 0,2 e 0,4. Isso indica que, conforme o número de restrições de recursos e máquinas aumenta, uma construção de solução mais orientada pela função gulosa é necessária para evitar caminhos aleatórios que resultem em sequenciamentos ineficientes.

Com base nessas observações, a TAB. 2 consolida os parâmetros definidos para o *GRASP*. Optou-se por uma parametrização escalonada que prioriza a agilidade em problemas simples e a qualidade da solução em cenários mais complexos.

Tabela 2 - Parâmetros definidos para o *GRASP*

Porte da instância	α	Iterações	Critério de escolha
Pequeno ($n_j = 20$)	0,8	$0,5 \times n_j$	Tempo
Médio ($n_j = 35$)	0,8	$1,5 \times n_j$	Qualidade e tempo
Grande ($n_j \geq 40$)	0,4	$2,5 \times n_j$	Qualidade e estabilidade
Geral	0,4	$2,5 \times n_j$	Robustez e confiabilidade

Fonte: Elaborada pelo autor (2026).

Embora algumas combinações tenham se destacado em instâncias específicas, a configuração geral foi adotada nos demais testes por apresentar bom desempenho em todos os cenários avaliados. Os parâmetros $\alpha = 0,4$ e número de iterações igual a $2,5 \times n_j$ produziram soluções de elevada qualidade de forma consistente, sendo definidos como configuração padrão do algoritmo *GRASP* neste trabalho.

5.7.3 Análise e definição dos parâmetros do *ILS*

Em continuidade à análise apresentada para o *GRASP*, o algoritmo *ILS* foi avaliado considerando os mesmos critérios de desempenho, *makespan* e tempo computacional. Os resultados indicam que seu comportamento também varia conforme a complexidade das instâncias, sendo necessária a definição de parâmetros que equilibrem qualidade da solução e tempo computacional.

Para gerar a solução inicial do *ILS*, foi utilizado o mesmo procedimento construtivo guloso-randomizado empregado no *GRASP*. Nesse processo, o valor de α não é sorteado, mas definido conforme os níveis estabelecidos no planejamento experimental. Esse parâmetro controla apenas a aleatoriedade da solução inicial, permitindo que o *ILS* comece as execuções com diferentes sequenciamentos. As perturbações realizadas posteriormente são independentes de α .

O nível de perturbação inicia em 1, correspondente à realização de duas trocas aleatórias entre posições da solução. A cada $1,5 \times n_j$ iterações consecutivas sem melhoria, uma nova troca é acrescentada, aumentando a intensidade da perturbação. Quando uma solução melhor é encontrada, o nível retorna ao valor inicial.

Para as instâncias de pequeno porte (I1 a I4), o algoritmo apresentou uma convergência acelerada. Assim como no *GRASP*, diferentes combinações de parâmetros resultaram no mesmo *makespan*, o que levou à escolha baseada no menor tempo computacional. À medida

que o tamanho das instâncias avançou para 35, 40 e 50 tarefas, os principais pontos observados foram:

- **Iterações:** Verificou-se que o desempenho do *ILS* está diretamente ligado ao número de iterações em problemas maiores. Enquanto instâncias menores foram resolvidas com poucas iterações, as instâncias de grande porte (I9 e I10) exigiram o nível de $2,5 \times nj$ para alcançar os melhores resultados de qualidade. Essa persistência na busca mostrou-se necessária para que o algoritmo pudesse refinar as soluções em espaços de busca mais amplos.
- **Aleatoriedade (Coeficiente α):** Diferente do observado no *GRASP*, o *ILS* apresentou desempenho superior com níveis de aleatoriedade mais altos. Nas instâncias médias, o valor de $\alpha = 0,8$ mostrou-se muito consistente. Já nas instâncias de maior porte, os melhores resultados de *makespan* foram obtidos com α entre 0,8 e 1,0.

Valores mais altos de α indicam que, para o *ILS*, uma exploração mais diversificada tanto no início da construção quanto nas fases de modificação da solução é benéfica. Na prática, esse maior nível de aleatoriedade amplia a diversificação da busca, favorecendo a exploração de diferentes regiões do espaço de soluções e reduzindo a ocorrência de estagnação em ótimos locais. Esse comportamento contribuiu para a obtenção de menores valores de *makespan* nas instâncias mais complexas.

Com base nessas observações, a TAB. 3 consolida os parâmetros definidos para o *ILS*. Optou-se por uma configuração geral de $\alpha = 0,8$ por ser o valor que apresentou maior estabilidade e repetição de bons resultados ao longo de todo o conjunto de testes.

Tabela 3 - Parâmetros definidos para o *ILS*

Porte da instância	α	Iterações	Critério de escolha
Pequeno ($nj = 20$)	0,8	$0,5 \times nj$	Tempo
Médio ($nj = 35$)	0,8	$1,5 \times nj$	Qualidade e tempo
Grande ($nj \geq 40$)	0,8	$2,5 \times nj$	Qualidade e estabilidade
Geral	0,8	$2,5 \times nj$	Robustez e confiabilidade

Fonte: Elaborada pelo autor (2026).

5.7.4 Análise e definição dos parâmetros do *VNS*

O algoritmo *VNS* foi avaliado sob a mesma ótica de desempenho. Por possuir uma dinâmica baseada na exploração sucessiva de diferentes estruturas de vizinhança, a análise concentrou-

se apenas em determinar o número de iterações necessárias para que a busca local dinâmica pudesse esgotar as possibilidades de melhoria sem comprometer excessivamente o tempo computacional.

Os resultados demonstraram que, diferente do *GRASP* e do *ILS*, o *VNS* apresenta uma dependência mais direta e linear entre o número de tarefas e o esforço de busca. Conforme observado nos experimentos fatoriais:

- Instâncias de Pequeno e Médio Porte: (I1 a I6): Para as instâncias de 20 tarefas, os coeficientes de $0,5 \times nj$ e $1 \times nj$ mostraram-se suficientes, atingindo o melhor *makespan* com baixo tempo computacional. Já nas instâncias de 35 tarefas (I5 e I6), começou a surgir uma distinção entre qualidade e equilíbrio: embora o melhor resultado absoluto tenha ocorrido com $2,5 \times nj$, o nível de $2 \times nj$ apresentou um desempenho muito próximo com um tempo de execução significativamente menor, destacando-se como uma escolha eficiente de equilíbrio.
- Instâncias de Grande Porte (I9 e I10): Nas instâncias de 50 tarefas, a necessidade de uma busca mais profunda tornou-se evidente. O nível de $2,5 \times nj$ foi consistentemente o responsável pelos menores *makespans* médios. Como o espaço de soluções para 50 tarefas é vasto, esse volume de iterações garantiu que a estratégia de embaralhar a ordem das vizinhanças ocorresse vezes o suficiente para diversificar a busca, evitando que o algoritmo encerrasse a execução precocemente em regiões de baixa qualidade.

Com base na análise realizada, a TAB. 4 consolida as escolhas finais, adotando o nível de $2,5 \times nj$ para a configuração geral, garantindo a consistência necessária para os cenários mais desafiadores do PMPIRR.

Tabela 4 – Parâmetros definidos para o *VNS*

Porte da Instância	Estratégia de vizinhança	Iterações	Critério de escolha
Pequeno ($nj = 20$)	Dinâmica (Sorteada)	$0,5 \times nj$	Tempo
Médio ($nj = 35$)	Dinâmica (Sorteada)	$2,0 \times nj$	Qualidade e tempo
Grande ($nj \geq 40$)	Dinâmica (Sorteada)	$2,5 \times nj$	Qualidade e estabilidade
Geral	Dinâmica (Sorteada)	$2,5 \times nj$	Robustez e confiabilidade

Fonte: Elaborada pelo autor (2026).

5.7.5 Análise e definição dos parâmetros do *Simulated Annealing*

Finalizando a etapa de parametrização, a análise dos resultados obtidos pelo algoritmo *Simulated Annealing* (SA) demonstrou um comportamento consistente ao longo das diferentes instâncias avaliadas. O desempenho foi analisado considerando o *makespan* obtido e o tempo computacional, buscando identificar a combinação de parâmetros capaz de fornecer soluções de melhor qualidade para diferentes portes do problema.

Para as instâncias de pequeno porte (I1 a I4, com 20 tarefas), observou-se que diferentes combinações de parâmetros produziram resultados muito próximos entre si. Nesses casos, a influência do número de iterações por temperatura (SA_{max}) foi reduzida, uma vez que o algoritmo conseguiu encontrar soluções de elevada qualidade mesmo com menores esforços computacionais. Além disso, verificou-se que as melhores configurações utilizaram valores de α próximos de 1, destacando o bom desempenho de taxas de resfriamento mais lentas.

À medida que o tamanho das instâncias aumentou para 35, 40 e 50 tarefas (I5 a I10), a influência do parâmetro SA_{max} tornou-se mais evidente. Nessas instâncias, a configuração correspondente a $2,5 \times nj$ apresentou os melhores resultados na maioria dos casos analisados. O aumento do número de iterações por temperatura permitiu uma exploração mais ampla do espaço de busca, favorecendo a obtenção de menores valores de *makespan*.

Em relação à taxa de resfriamento, observou-se que o valor $\alpha = 0,99$ apresentou o melhor desempenho na maior parte das instâncias avaliadas. Esse comportamento indica que um resfriamento mais gradual permite ao algoritmo explorar por mais tempo diferentes regiões do espaço de soluções antes da convergência, reduzindo a probabilidade de aprisionamento em ótimos locais.

Com base nesses resultados, a TAB. 5 apresenta os parâmetros definidos para o algoritmo *Simulated Annealing*. Embora algumas instâncias de pequeno porte tenham apresentado desempenho semelhante para diferentes configurações, a combinação $\alpha = 0,99$ e $SA_{max} = 2,5 \cdot nj$ destacou-se por apresentar bons resultados em todos os portes de instância avaliados. Dessa forma, essa configuração foi adotada como padrão nos experimentos computacionais subsequentes deste trabalho.

Tabela 5 - Parâmetros definidos para o *SA*

Porte da Instância	α	Iterações	Critério de escolha
Pequeno ($n_j = 20$)	0,99	$1,0 \times n_j$	Tempo
Médio ($n_j = 35$)	0,99	$2,5 \times n_j$	Qualidade
Grande ($n_j \geq 40$)	0,99	$2,5 \times n_j$	Qualidade
Geral	0,99	$2,5 \times n_j$	Desempenho geral

Fonte: Elaborada pelo autor (2026).

5.7.6 Parâmetros adotados nos experimentos computacionais

Portanto, com base nos resultados obtidos durante a calibração, foram definidos os parâmetros utilizados nos experimentos computacionais. A TAB. 6 apresenta as configurações finais adotadas para cada meta-heurística.

Tabela 6 – Parâmetros definidos para as meta-heurísticas

Meta-heurística	Parâmetros
<i>GRASP</i>	$\alpha = 0,4$; Iterações = $2,5 \times n_j$
<i>ILS</i>	$\alpha = 0,8$; Iterações = $2,5 \times n_j$
<i>VNS</i>	Vizinhanças dinâmicas (sorteadas); Iterações = $2,5 \times n_j$
<i>SA</i>	$\alpha = 0,99$; $S_{Amax} = 2,5 \times n_j$; $Tf = 0,001$; $T0$ adaptativa

Fonte: Elaborada pelo autor (2026).

6 RESULTADOS E DISCUSSÃO

Para avaliação dos resultados, utilizou-se um limite inferior simplificado calculado pela divisão da soma total dos tempos de processamento pelo número de máquinas disponíveis, desconsiderando as restrições de recursos e recursos compartilhados. Dessa forma, o limite inferior foi utilizado apenas como uma referência aproximada da melhor condição teórica possível para cada instância. O limite inferior adotado é apresentado na Equação 3.

$$LB = \frac{\sum_{j=1}^n p_j}{m} \quad (3)$$

Em que LB representa o limite inferior, p_j corresponde ao tempo de processamento da tarefa j , n é o número total de tarefas e m o número de máquinas disponíveis.

A partir do limite inferior, foram calculados o desvio absoluto (d) e o desvio percentual (dp) das soluções obtidas. O desvio absoluto representa a diferença, em unidades de tempo, entre o *makespan* encontrado pelo algoritmo e o respectivo limite inferior, conforme a Equação 4.

$$d = C_{max} - LB \quad (4)$$

Também foi calculado o desvio percentual, que indica quanto a solução ficou acima do limite inferior em termos proporcionais, e não apenas em unidades de tempo. Essa medida foi calculada conforme a Equação 5.

$$dp(\%) = \frac{C_{max} - LB}{LB} \times 100 \quad (5)$$

Em que dp representa o desvio percentual em relação ao limite inferior. Quanto menores os desvios absoluto e percentual, mais próxima a solução está do limite adotado. Quando $d = 0$ e, consequentemente, $dp = 0\%$, o *makespan* obtido coincide com o limite inferior, permitindo concluir que a solução é ótima.

Durante os testes preliminares, observou-se que algumas instâncias com 100 tarefas demandavam tempos de processamento superiores a 11.000 segundos para determinadas configurações das meta-heurísticas. Embora esse tempo possa ser aceitável em alguns contextos, ele tornaria a realização das 30 execuções previstas excessivamente demorada.

Dessa forma, foi estabelecido um limite máximo de execução de 300 segundos para as instâncias com 100 tarefas e de 600 segundos para as instâncias com 150 tarefas. O critério de

parada era verificado ao término de cada iteração, por isso, uma iteração iniciada antes do limite era concluída, fazendo com que o tempo total pudesse ultrapassar o valor estabelecido. Com isso, a adoção desses limites evitou execuções excessivamente longas e tornou possível realizar as 30 execuções previstas para cada meta-heurística em cada instância.

6.1 Testes com instâncias da literatura

Após a etapa de definição e calibração dos parâmetros, realizou-se a validação das configurações selecionadas em diferentes conjuntos de instâncias. Inicialmente, os algoritmos foram avaliados utilizando instâncias clássicas da literatura, permitindo verificar o comportamento das meta-heurísticas em cenários amplamente utilizados em trabalhos correlatos.

Nesta seção, os parâmetros definidos na etapa anterior são avaliados por meio de sua aplicação em instâncias clássicas da literatura propostas por Chung *et al.* (2019). Para os experimentos, foi selecionada uma instância representativa (HA001) para cada combinação de número de tarefas, máquinas e recursos, considerando tempos de processamento no intervalo de 1 a 100.

Cada algoritmo (*GRASP*, *ILS*, *VNS* e *SA*) foi executado 30 vezes para cada cenário, permitindo a análise de métricas como *makespan* médio, variabilidade das soluções (variância e desvio padrão), desvios absoluto e percentual em relação ao limite inferior e tempo computacional médio.

Os resultados obtidos demonstraram que os quatro algoritmos encontraram soluções de elevada qualidade para todas as instâncias avaliadas. Conforme apresentado na TAB. 7, em grande parte dos casos, os valores médios de *makespan* coincidiram com os respectivos limites inferiores, resultando em desvios percentuais médios nulos. Nas demais instâncias, os desvios permaneceram muito próximos de zero, sendo o maior valor observado igual a apenas 0,084%. Nas instâncias em que o *makespan* coincidiu com o limite inferior, a otimalidade da solução foi comprovada. Nas demais, os baixos desvios observados indicam soluções de elevada qualidade.

Tabela 7 - Desvio percentual médio das meta-heurísticas nas instâncias da literatura

Instância	GRASP (%)	ILS (%)	VNS (%)	SA (%)
20-2-4	0,084	0,084	0,084	0,084
20-2-6	0,000	0,000	0,000	0,000
20-2-8	0,000	0,000	0,000	0,000
20-2-10	0,000	0,000	0,000	0,000
50-2-4	0,000	0,000	0,000	0,000
50-2-6	0,046	0,046	0,046	0,046
50-2-8	0,000	0,000	0,000	0,000
50-2-10	0,000	0,000	0,000	0,000
100-2-4	0,019	0,019	0,019	0,019
100-2-6	0,020	0,020	0,020	0,020
100-2-8	0,000	0,000	0,000	0,000
100-2-10	0,000	0,000	0,000	0,000
150-2-4	0,032	0,051	0,016	0,013
150-2-6	0,000	0,000	0,002	0,000
150-2-8	0,014	0,014	0,014	0,014
150-2-10	0,000	0,000	0,000	0,000

Fonte: Elaborada pelo autor (2026).

Embora todos os métodos tenham produzido soluções de qualidade semelhante para as instâncias da literatura, diferenças significativas foram observadas em relação ao tempo computacional. A TAB. 8 apresenta o tempo médio de execução de cada método para os diferentes portes de instância avaliados.

Tabela 8 - Tempo computacional médio das meta-heurísticas por porte de instância

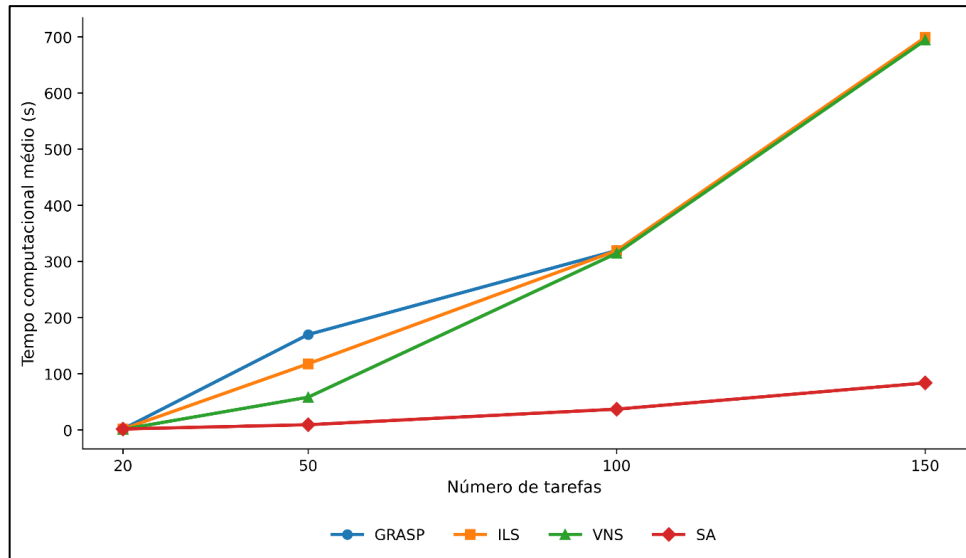
Nº de tarefas	GRASP (s)	ILS (s)	VNS (s)	SA (s)
20	1,50	1,43	1,02	1,60
50	169,77	117,70	58,20	9,17
100	319,49	319,27	314,55	36,89
150	697,24	699,16	694,14	83,56

Fonte: Elaborada pelo autor (2026).

Observa-se que, para as instâncias de pequeno porte (20 tarefas), todos os métodos apresentaram tempos de execução inferiores a 3 segundos. Entretanto, à medida que o tamanho das instâncias aumentou, as diferenças de desempenho tornaram-se mais evidentes. O algoritmo *Simulated Annealing* destacou-se por apresentar os menores tempos médios de execução em todos os grupos de instâncias com 50, 100 e 150 tarefas, mantendo a mesma qualidade das soluções obtidas pelos demais métodos.

A FIG. 11 ilustra o comportamento do tempo computacional médio em função do número de tarefas das instâncias. Nota-se que o crescimento do tempo de execução foi significativamente mais acentuado para os algoritmos *GRASP*, *ILS* e *VNS*, enquanto o *Simulated Annealing* apresentou uma taxa de crescimento consideravelmente menor.

Figura 11 – Tempo computacional médio das meta-heurísticas em função do número de tarefas das instâncias de Chung.



Fonte: Elaborada pelo autor (2026).

De modo geral, os resultados demonstram que todas as abordagens encontraram soluções ótimas ou muito próximas do ótimo para as instâncias avaliadas. Contudo, considerando simultaneamente a qualidade das soluções e o esforço computacional requerido, o *Simulated Annealing* apresentou o melhor desempenho global, especialmente para instâncias de maior porte, nas quais obteve soluções equivalentes às demais meta-heurísticas em tempos significativamente inferiores.

6.2 Testes com instâncias geradas para o trabalho

Após a validação das meta-heurísticas utilizando as instâncias propostas por Chung *et al.* (2019), realizou-se uma segunda etapa de experimentos empregando instâncias desenvolvidas especificamente para este trabalho. Diferentemente das instâncias da literatura, restritas ao caso de duas máquinas paralelas idênticas, as novas instâncias foram geradas considerando diferentes quantidades de máquinas, permitindo avaliar o comportamento dos algoritmos em um problema mais geral e em cenários de maior complexidade.

6.2.1 Geração das instâncias

Para gerar as instâncias, foi desenvolvido um código em *Python*, elaborado com o auxílio do *ChatGPT*, capaz de produzir automaticamente diferentes configurações do problema a partir da definição do número de tarefas, máquinas e recursos compartilhados. O gerador cria, de forma aleatória, os tempos de processamento das tarefas, distribuídos uniformemente no intervalo de 1 a 100 unidades de tempo, e define aleatoriamente o recurso utilizado por cada tarefa.

Foram geradas 24 instâncias contemplando diferentes combinações entre número de tarefas, máquinas e recursos compartilhados, possibilitando avaliar o desempenho das abordagens propostas em problemas com diferentes configurações e níveis de complexidade. A TAB. 9 apresenta as características das instâncias geradas.

Tabela 9 – Configuração das instâncias geradas para o trabalho

Instância	Nº de tarefas	Nº de máquinas	Nº de recursos
20-2-2	20	2	2
20-2-4	20	2	4
20-4-2	20	4	2
20-4-4	20	4	4
20-8-2	20	8	2
20-8-4	20	8	4
50-2-2	50	2	2
50-2-4	50	2	4
50-4-2	50	4	2
50-4-4	50	4	4
50-8-2	50	8	2
50-8-4	50	8	4
100-2-2	100	2	2
100-2-4	100	2	4
100-4-2	100	4	2
100-4-4	100	4	4
100-8-2	100	8	2
100-8-4	100	8	4
150-2-2	150	2	2
150-2-4	150	2	4
150-4-2	150	4	2
150-4-4	150	4	4
150-8-2	150	8	2
150-8-4	150	8	4

Fonte: Elaborada pelo autor (2026).

Na TAB. 9, a nomenclatura utilizada para identificar cada instância segue o formato T-M-R, em que T representa o número de tarefas (*jobs*), M o número de máquinas e R o número de recursos

compartilhados. Assim, por exemplo, a instância 20-2-2 corresponde a um problema composto por 20 tarefas, 2 máquinas e 2 recursos, enquanto a instância 150-8-4 representa um cenário com 150 tarefas, 8 máquinas e 4 recursos.

6.2.2 Análise das instâncias de pequeno porte

Inicialmente, foram analisadas as instâncias de pequeno porte, compostas por 20 tarefas. A TAB. 10 apresenta os desvios percentuais médios obtidos pelo *GRASP*, *ILS*, *VNS* e *Simulated Annealing* para as seis configurações avaliadas.

Tabela 10 – Desvio percentual médio das meta-heurísticas para as instâncias de 20 tarefas

Instância	GRASP (%)	ILS (%)	VNS (%)	SA (%)
20-2-2	4,85	4,85	4,85	4,85
20-2-4	0,09	0,09	0,09	0,09
20-4-2	106,25	106,25	106,25	106,25
20-4-4	88,29	88,29	88,29	88,33
20-8-2	309,37	309,37	309,37	309,37
20-8-4	187,48	185,12	185,12	188,87

Fonte: Elaborada pelo autor (2026).

Nota-se que, nas instâncias 20-2-2, 20-2-4, 20-4-2 e 20-8-2, todos os algoritmos apresentaram os mesmos desvios percentuais médios, correspondentes a 4,85%, 0,09%, 106,25% e 309,37%, respectivamente. Esse resultado indica que, nessas configurações, os algoritmos apresentaram desempenho equivalente em termos de qualidade das soluções.

Na instância 20-4-4, *GRASP*, *ILS* e *VNS* obtiveram desvio percentual médio de 88,29%, enquanto o *SA* apresentou 88,33%, diferença praticamente desprezível. Já na instância 20-8-4, foram observadas diferenças mais perceptíveis: *ILS* e *VNS* apresentaram o menor desvio percentual médio, de 185,12%, seguidos pelo *GRASP*, com 187,48%, e pelo *SA*, com 188,87%.

Além da qualidade das soluções, também foi analisado o desempenho computacional. A TAB. 11 apresenta o tempo computacional médio das meta-heurísticas para as instâncias de 20 tarefas.

Tabela 11 – Tempo computacional médio em (s) das meta-heurísticas para as instâncias de 20 tarefas

Instância	<i>GRASP</i>	<i>ILS</i>	<i>VNS</i>	<i>SA</i>
20-2-2	3,32	2,38	1,19	1,65
20-2-4	2,51	1,99	1,17	1,64
20-4-2	4,33	2,50	1,34	1,79
20-4-4	5,09	2,93	1,63	1,89
20-8-2	8,34	3,46	1,99	1,94
20-8-4	14,64	5,12	3,60	2,00

Fonte: Elaborada pelo autor (2026).

Verifica-se que todos os métodos resolveram as instâncias de pequeno porte em tempos computacionais reduzidos, inferiores a 15 segundos. Entretanto, diferenças significativas podem ser observadas entre os algoritmos. O *GRASP* apresentou os maiores tempos médios em todas as instâncias avaliadas. Esse aumento foi mais acentuado à medida que cresceu o número de máquinas disponíveis, indicando maior esforço computacional durante a construção e refinamento das soluções.

O *ILS* apresentou desempenho intermediário, com tempos variando entre 1,99 segundos e 5,12 segundos, mantendo crescimento gradual conforme a complexidade das instâncias aumentou. O *VNS* destacou-se como o método mais rápido na maior parte das instâncias, registrando tempos entre 1,17 segundos e 3,60 segundos. O *Simulated Annealing* apresentou desempenho semelhante ao *VNS* nas cinco primeiras configurações e obteve o menor tempo computacional na instância 20-8-4, concluindo sua execução em apenas 2,00 segundos.

De maneira geral, verifica-se que o aumento do número de máquinas e recursos elevou o tempo de execução de todos os algoritmos, porém esse crescimento foi significativamente mais pronunciado para o *GRASP*. Em contrapartida, o *VNS* e o *Simulated Annealing* mantiveram tempos computacionais reduzidos em todas as configurações avaliadas, evidenciando maior eficiência computacional sem comprometer a qualidade das soluções obtidas.

Também foi avaliada a estabilidade das soluções ao longo das 30 execuções realizadas para cada instância. De modo geral, observou-se baixa variabilidade nos resultados, indicando que todos os algoritmos produziram soluções consistentes para as instâncias de pequeno porte. A TAB. 12 apresenta o desvio-padrão do *makespan* obtido por cada meta-heurística.

Tabela 12 – Desvio-padrão do *makespan* das meta-heurísticas para as instâncias de 20 tarefas

Instância	GRASP	ILS	VNS	SA
20-2-2	0,00	0,00	0,00	0,00
20-2-4	0,00	0,00	0,00	0,00
20-4-2	0,00	0,00	0,00	0,00
20-4-4	0,00	0,00	0,00	0,40
20-8-2	0,00	0,00	0,00	0,00
20-8-4	5,86	0,00	0,00	8,74

Fonte: Elaborada pelo autor (2026).

Os resultados mostram que para praticamente todas as instâncias, o desvio-padrão foi igual a zero, evidenciando elevada estabilidade das soluções obtidas pelas meta-heurísticas. As exceções ocorreram nas instâncias 20-4-4 e 20-8-4, nas quais o *Simulated Annealing* apresentou pequenas variações no *makespan*. Adicionalmente, na instância 20-8-4, o *GRASP* também apresentou variabilidade, ao passo que o *ILS* e o *VNS* mantiveram o mesmo *makespan* em todas as execuções.

6.2.3 Análise das instâncias de médio porte

As instâncias de médio porte, compostas por 50 tarefas, apresentaram diferenças mais expressivas entre os métodos quando comparadas às instâncias de pequeno porte. A TAB. 13 apresenta o desvio percentual médio, calculado conforme a Equação 5, permitindo avaliar, em termos proporcionais, quanto o *makespan* obtido por cada meta-heurística ficou acima do respectivo limite inferior nas seis instâncias avaliadas.

Tabela 13 – Desvio percentual médio obtido pelas meta-heurísticas para as instâncias de 50 tarefas

Instância	GRASP (%)	ILS (%)	VNS (%)	SA (%)
50-2-2	3,47	3,47	3,47	3,47
50-2-4	0,04	0,04	0,04	0,04
50-4-2	127,06	127,06	127,06	127,06
50-4-4	35,70	26,87	27,00	25,38
50-8-2	319,48	319,48	319,48	319,48
50-8-4	164,46	149,57	151,00	148,89

Fonte: Elaborada pelo autor (2026).

Observa-se que, nas instâncias 50-2-2, 50-2-4, 50-4-2 e 50-8-2, todas as abordagens apresentaram exatamente o mesmo desvio percentual médio, indicando que os algoritmos obtiveram soluções com a mesma qualidade média em relação ao limite inferior.

Entretanto, nas instâncias 50-4-4 e 50-8-4 passaram a surgir diferenças mais expressivas. Em ambas as configurações, o *Simulated Annealing* apresentou os menores desvios percentuais médios, seguido pelo *ILS*, *VNS* e *GRASP*, evidenciando sua maior capacidade de encontrar soluções próximas ao limite inferior.

Além de apresentar os menores desvios percentuais médios nas instâncias em que houve diferença entre as meta-heurísticas, o *Simulated Annealing* também foi o algoritmo com melhor desempenho computacional. A TAB. 14 apresenta o tempo computacional médio dos métodos para as instâncias de médio porte, evidenciando que o SA manteve tempos inferiores a 12 segundos em todas as configurações avaliadas, enquanto os demais algoritmos apresentaram crescimento expressivo do tempo de execução.

Tabela 14 – Tempo computacional médio (s) das meta-heurísticas para as instâncias de 50 tarefas

Instância	<i>GRASP</i>	<i>ILS</i>	<i>VNS</i>	<i>SA</i>
50-2-2	270,77	200,11	73,04	9,45
50-2-4	363,49	286,96	128,19	8,82
50-4-2	337,33	169,20	70,38	10,27
50-4-4	859,62	608,16	319,26	10,68
50-8-2	315,53	193,31	78,97	11,16
50-8-4	1010,31	670,18	381,62	11,48

Fonte: Elaborada pelo autor (2026).

Nota-se que o *GRASP* apresentou os maiores tempos computacionais em todas as configurações avaliadas, seguido pelo *ILS* e pelo *VNS*. Em contrapartida, o *Simulated Annealing* manteve tempo de execução praticamente constante, inferior a 12 segundos em todas as instâncias, demonstrando baixa sensibilidade às variações no número de máquinas e recursos compartilhados. Assim, além de apresentar os menores desvios percentuais médios nas instâncias em que houve diferença entre os algoritmos, o *Simulated Annealing* também se destacou como a meta-heurística mais eficiente do ponto de vista computacional.

Quanto à estabilidade das soluções, a TAB. 15 apresenta o desvio-padrão do desvio absoluto obtido por cada meta-heurística para as instâncias de médio porte.

Tabela 15 – Desvio-padrão do desvio absoluto das meta-heurísticas para as instâncias de 50 tarefas

Instância	GRASP	ILS	VNS	SA
50-2-2	0,00	0,00	0,00	0,00
50-2-4	0,00	0,00	0,00	0,00
50-4-2	0,00	0,00	0,00	0,00
50-4-4	33,32	34,64	32,63	25,29
50-8-2	0,00	0,00	0,00	0,00
50-8-4	17,83	23,75	23,20	20,33

Fonte: Elaborada pelo autor (2026).

A análise da TAB. 15 mostra que, assim como nas instâncias de pequeno porte, as configurações 50-2-2, 50-2-4, 50-4-2 e 50-8-2 apresentaram desvio-padrão nulo para todas as abordagens, indicando elevada estabilidade das soluções obtidas. Nas instâncias 50-4-4 e 50-8-4, verificou-se variabilidade entre as execuções, sendo que o *Simulated Annealing* apresentou o menor desvio-padrão na instância 50-4-4, enquanto o *GRASP* apresentou o menor desvio-padrão na instância 50-8-4. De modo geral, os resultados evidenciam que todas as meta-heurísticas mantiveram comportamento consistente, mesmo nas configurações em que foram observadas diferenças na qualidade das soluções.

6.2.4 Análise das instâncias de grande porte

Por fim, são apresentados os resultados obtidos para as instâncias de grande porte, compostas por 100 e 150 tarefas. Esses cenários representam os casos de maior complexidade avaliados neste trabalho, permitindo analisar o comportamento dos métodos em problemas de maior dimensão.

Inicialmente, foi avaliada a qualidade das soluções obtidas. A TAB. 16 apresenta o desvio percentual médio das meta-heurísticas para todas as instâncias de grande porte.

Tabela 16 – Desvio percentual médio das meta-heurísticas para as instâncias de 100 e 150 tarefas

Instância	GRASP (%)	ILS (%)	VNS (%)	SA (%)
100-2-2	21,739	21,549	21,549	21,549
100-2-4	0,001	0,004	0,000	0,000
100-4-2	105,760	106,951	107,870	100,867
100-4-4	63,642	58,128	56,014	28,612
100-8-2	329,088	311,262	313,026	309,080
100-8-4	234,329	223,149	225,456	180,063
150-2-2	23,839	15,462	17,516	1,455
150-2-4	0,022	0,019	0,027	0,000
150-4-2	149,535	136,697	138,694	116,824
150-4-4	77,328	78,264	75,371	41,603
150-8-2	402,333	363,386	354,091	341,487
150-8-4	257,803	247,731	252,820	165,650

Fonte: Elaborada pelo autor (2026).

Observa-se que, nas instâncias 100-2-2 e 100-2-4, todas as abordagens apresentaram desempenho praticamente equivalente, obtendo desvios percentuais médios muito próximos entre si. Comportamento semelhante foi observado na instância 150-2-4, na qual todos os algoritmos produziram soluções praticamente idênticas e com desvio percentual próximo de zero.

Nas demais configurações, as diferenças entre os métodos tornaram-se mais evidentes. O *Simulated Annealing* apresentou os menores desvios percentuais médios em todas essas instâncias, destacando-se principalmente nas configurações 100-4-4, 100-8-4, 150-2-2, 150-4-4 e 150-8-4, nas quais a redução do desvio percentual em relação aos demais métodos foi mais significativa. De modo geral, o *ILS* e o *VNS* apresentaram desempenhos semelhantes entre si, enquanto o *GRASP* obteve os maiores desvios percentuais médios na maior parte das instâncias avaliadas.

Esses resultados evidenciam que o aumento da dimensão do problema ampliou as diferenças entre as estratégias de busca empregadas pelas meta-heurísticas. Enquanto nas configurações mais simples os algoritmos obtiveram soluções semelhantes, nas instâncias mais complexas o *Simulated Annealing* apresentou maior capacidade de encontrar soluções com menores desvios percentuais em relação ao limite inferior.

A TAB. 17 apresenta o tempo computacional médio dos métodos para as instâncias de grande porte. Para viabilizar a realização dos experimentos, foi estabelecido um limite máximo de execução de 300 segundos para as instâncias com 100 tarefas e de 600 segundos para as

instâncias com 150 tarefas. Essa limitação foi adotada porque, nos testes iniciais, as execuções das instâncias de 100 tarefas demandaram aproximadamente 11.000 segundos, tornando inviável a realização das 30 execuções previstas para cada configuração.

Tabela 17 – Tempo computacional médio em (s) das meta-heurísticas para as instâncias de 100 e 150 tarefas

Instância	<i>GRASP</i>	<i>ILS</i>	<i>VNS</i>	<i>SA</i>
100-2-2	320,90	320,51	316,13	40,07
100-2-4	326,50	324,38	318,59	38,80
100-4-2	313,49	318,53	314,05	41,81
100-4-4	334,94	324,47	315,73	43,28
100-8-2	313,87	315,04	314,98	44,19
100-8-4	338,23	329,55	318,54	44,84
150-2-2	670,48	694,59	650,85	86,08
150-2-4	760,29	762,19	714,95	85,63
150-4-2	698,03	664,08	643,41	97,99
150-4-4	787,91	729,85	697,29	99,67
150-8-2	674,19	694,00	648,20	101,79
150-8-4	736,56	746,99	735,94	110,00

Fonte: Elaborada pelo autor (2026).

Assim, os tempos observados para o *GRASP*, *ILS* e *VNS* nas instâncias de 100 e 150 tarefas refletem o limite de execução adotado, uma vez que esses métodos foram interrompidos antes de atingir seu critério natural de parada. Já o *Simulated Annealing* finalizou suas execuções pelo próprio critério de parada do algoritmo.

Os resultados mostram que o aumento do número de tarefas elevou o tempo de execução de todas as meta-heurísticas. Os tempos médios do *GRASP*, *ILS* e *VNS* permaneceram próximos ou acima dos critérios de parada estabelecidos, em razão da conclusão das iterações que já estavam em andamento. Em contrapartida, o *Simulated Annealing* manteve tempos computacionais substancialmente inferiores, variando entre 38,80 segundos e 44,84 segundos para as instâncias de 100 tarefas e entre 85,63 segundos e 110,00 segundos para as instâncias de 150 tarefas. Esse comportamento indica que o crescimento da dimensão do problema afetou de forma muito menos intensa o desempenho computacional do *Simulated Annealing* quando comparado às demais meta-heurísticas.

Quanto à estabilidade das soluções, a TAB. 18 apresenta o desvio-padrão do desvio absoluto obtido para as instâncias de grande porte.

Tabela 18 – Desvio-padrão do desvio absoluto das meta-heurísticas para as instâncias de 100 e 150 tarefas

Instância	GRASP	ILS	VNS	SA
100-2-2	17,49	0,00	0,00	0,00
100-2-4	0,18	0,31	0,00	0,00
100-4-2	106,06	183,64	240,17	0,00
100-4-4	96,32	128,19	107,45	36,15
100-8-2	215,71	71,71	103,68	0,00
100-8-4	122,13	137,53	111,16	41,96
150-2-2	439,32	598,65	517,94	0,00
150-2-4	3,15	3,65	4,94	0,00
150-4-2	504,36	441,20	595,57	0,00
150-4-4	176,54	234,20	212,73	64,32
150-8-2	573,73	470,75	304,17	0,00
150-8-4	144,27	186,70	208,86	58,01

Fonte: Elaborada pelo autor (2026).

Verifica-se que o *Simulated Annealing* apresentou elevada estabilidade, obtendo desvio-padrão igual a zero em oito das doze instâncias avaliadas e os menores valores de variabilidade na maior parte das demais configurações. O comportamento do *GRASP*, *ILS* e *VNS* variou conforme a instância, não sendo observada uma predominância clara entre esses três métodos. Esses resultados mostram que o *Simulated Annealing*, além de apresentar menores desvios percentuais médios e menor tempo computacional na maioria das configurações, também produziu resultados mais consistentes entre as diferentes execuções.

6.2.5 Análise das replicações da instância 150-2-2

Com o objetivo de ilustrar o comportamento dos métodos em uma instância de elevada complexidade, foi selecionada a instância 150-2-2, por apresentar uma das maiores diferenças de desempenho entre os algoritmos. A TAB. 19 resume os desvios percentuais obtidos nas 30 replicações realizadas para cada meta-heurística.

Tabela 19 – Resumo dos desvios percentuais nas replicações da instância 150-2-2

Método	Menor desvio percentual (%)	Maior desvio percentual (%)	Desvio percentual (%) médio	Replicações com o menor desvio percentual
GRASP	1,455	48,729	23,839	3/30
ILS	1,455	70,836	15,462	13/30
VNS	1,455	58,606	17,516	8/30
SA	1,455	1,455	1,455	30/30

Fonte: Elaborada pelo autor (2026).

A TAB. 19 mostra que todas as meta-heurísticas atingiram o menor desvio percentual observado, de 1,455%, em pelo menos uma replicação. Entretanto, apenas o *Simulated Annealing* alcançou esse resultado em todas as 30 execuções. Por outro lado, *GRASP*, *ILS* e *VNS* apresentaram maior variação entre as replicações, atingindo desvios percentuais máximos de 48,729%, 70,836% e 58,606%, respectivamente.

Esses resultados mostram que o desempenho do *SA* não decorreu de uma execução isolada, pois o algoritmo produziu o mesmo resultado em todas as replicações. Além disso, o *SA* apresentou o menor desvio percentual médio e ausência de variação entre as soluções encontradas nessa instância. Esse comportamento reforça os resultados das seções anteriores, nas quais o *Simulated Annealing* apresentou menores desvios percentuais médios, menor tempo computacional e maior estabilidade entre as meta-heurísticas avaliadas.

6.3 Comparação com o modelo de programação linear resolvido pelo *Gurobi*

Com o objetivo de avaliar a qualidade das soluções obtidas pelas meta-heurísticas, foi utilizado o *Gurobi Optimizer* para resolver um modelo de Programação Linear Inteira Mista (PLIM) para o problema de máquinas paralelas idênticas com restrição de recursos compartilhados. O modelo teve como objetivo minimizar o *makespan* (C_{max}).

O modelo matemático foi desenvolvido com base na formulação proposta por Chung *et al.* (2019), originalmente elaborada para o problema de duas máquinas paralelas idênticas com restrição de moldes. Neste trabalho, essa formulação foi adaptada para considerar um número arbitrário de máquinas paralelas, tornando-a compatível com o problema estudado nesta pesquisa. A implementação computacional foi realizada na linguagem *Python*, utilizando a biblioteca *gurobipy*, com o auxílio do *ChatGPT* como ferramenta de apoio na geração do código.

O modelo considera um conjunto de tarefas J , máquinas paralelas idênticas e recursos compartilhados. Cada tarefa possui um tempo de processamento p_j e está associada a um recurso específico. Dessa forma, tarefas que utilizam o mesmo recurso não podem ser processadas simultaneamente, mesmo que existam máquinas disponíveis.

Embora o modelo tenha sido adaptado para um número arbitrário de máquinas, optou-se por utilizá-lo apenas nas instâncias de 20 tarefas. Para as instâncias de maior porte, o elevado crescimento do espaço de busca tornou inviável a obtenção da solução ótima em tempo computacional adequado. Assim, foi estabelecido um limite de 1800 segundos por execução.

Quando o *Gurobi* encontrava a solução ótima dentro desse limite, seus resultados eram utilizados como referência para comparação. Nos casos em que a otimalidade não foi comprovada dentro do limite de tempo, foi considerada para comparação a melhor solução viável encontrada pelo *Gurobi* até o encerramento da execução.

A função objetivo do modelo consiste em minimizar o *makespan* (C_{max}), conforme apresentado na Equação (6).

$$\min C_{max} \quad (6)$$

A primeira modificação realizada em relação ao modelo original de Chung *et al.* (2019) corresponde à Equação (7), apresentada a seguir.

$$\sum_{h=1}^n y_h \leq m \quad (7)$$

Em que y_h é uma variável binária que assume valor 1 quando a tarefa h inicia a sequência de processamento em uma máquina e 0 caso contrário, enquanto m representa o número de máquinas paralelas disponíveis.

Além disso, x_{ih} indica se a tarefa i é processada imediatamente antes da tarefa h na mesma máquina, b_{ih} representa a relação de precedência entre as tarefas, C_h corresponde ao tempo de conclusão da tarefa h , M é uma constante suficientemente grande, definida neste trabalho como a soma dos tempos de processamento de todas as tarefas acrescida do maior tempo de processamento da instância e z_{ih} é um parâmetro que identifica se as tarefas i e h utilizam o mesmo recurso.

As demais restrições do modelo são apresentadas nas Equações (8) a (19), com base na formulação de Chung *et al.* (2019). Essas restrições foram mantidas conforme o modelo original, com exceção da Equação (15), na qual o limite de duas tarefas também foi substituído por m , adequando a formulação ao número de máquinas considerado neste trabalho.

$$y_h + \sum_{i=1, i \neq h}^n x_{ih} = 1, \quad h = 1, \dots, n \quad (8)$$

$$\sum_{i=1, i \neq h}^{n+1} x_{hi} = 1, \quad h = 1, \dots, n \quad (9)$$

$$C_h - p_h y_h \geq 0, \quad h = 1, \dots, n \quad (10)$$

$$b_{hi} + b_{ih} = 1, \quad i, h = 1, \dots, n, \quad i \neq h \quad (11)$$

$$\sum_{i=1, i \neq h}^n (b_{hi} + b_{ih}) + 1 = n, \quad h = 1, \dots, n-1 \quad (12)$$

$$C_h - C_i + M(1 - b_{ih}) + M z_{ih} \geq p_h, \quad i, h = 1, \dots, n, \quad i \neq h \quad (13)$$

$$C_h - C_i + M(1 - x_{ih}) \geq p_h, \quad i, h = 1, \dots, n, \quad i \neq h \quad (14)$$

$$\sum_{i=1}^n x_{i,n+1} \leq m \quad (15)$$

$$x_{ih} + x_{hi} \leq 1, \quad i, h = 1, \dots, n, \quad i \neq h \quad (16)$$

$$C_{max} - C_h \geq 0, \quad h = 1, \dots, n \quad (17)$$

$$b_{ih}, x_{ih}, y_h \in \{0,1\} \quad (18)$$

$$C_{max}, C_h \geq 0 \quad (19)$$

As Equações (8) e (9) definem a posição das tarefas nas sequências de processamento, enquanto a Equação (10) relaciona o tempo de conclusão da primeira tarefa ao seu tempo de processamento. As Equações (11) e (12) estabelecem as relações de precedência entre as tarefas. Já as Equações (13) e (14) garantem o correto sequenciamento das tarefas, considerando tanto a utilização dos recursos compartilhados quanto a ordem de processamento em uma mesma máquina.

A Equação (15) o número de tarefas que podem encerrar as sequências de processamento ao número de máquinas disponíveis, enquanto a Equação (16) estabelece a consistência da ordem

entre as tarefas. A Equação (17) define o *makespan* do problema e, por fim, as Equações (18) e (19) estabelecem o domínio das variáveis utilizadas no modelo.

A TAB. 20 apresenta a comparação entre os resultados obtidos pelo modelo matemático resolvido pelo *Gurobi* e pelas meta-heurísticas para as instâncias de 20 tarefas.

Tabela 20 – Comparação entre as soluções obtidas pelo *Gurobi* e pelas meta-heurísticas para as instâncias de 20 tarefas

Instância	<i>Gurobi</i>	Status	<i>GRASP</i>	<i>ILS</i>	<i>VNS</i>	<i>SA</i>
20-2-2	627	Tempo	627	627	627	627
20-2-4	573	Tempo	573	573	573	573
20-4-2	429	Ótimo	429	429	429	429
20-4-4	434	Ótimo	434	434	434	434
20-8-2	414	Tempo	414	414	414	414
20-8-4	314	Ótimo	314	314	314	314

Fonte: Elaborada pelo autor (2026).

Verifica-se que todas as meta-heurísticas obtiveram o mesmo *makespan* encontrado pelo *Gurobi* em todas as instâncias avaliadas. Nas instâncias em que o solucionador comprovou a otimalidade, esse resultado confirma que os algoritmos também encontraram a solução ótima. Já nas instâncias em que o limite de tempo foi atingido, as meta-heurísticas alcançaram a mesma melhor solução encontrada pelo *Gurobi*, evidenciando a elevada qualidade das soluções produzidas.

Além da qualidade das soluções, também foi comparado o tempo computacional necessário para obtenção dos resultados. A TAB. 21 apresenta o tempo de execução do *Gurobi* e o tempo computacional médio das meta-heurísticas para as instâncias de 20 tarefas.

Tabela 21 – Tempo computacional em (s) do *Gurobi* e das meta-heurísticas para as instâncias de 20 tarefas

Instância	<i>Gurobi</i>	<i>GRASP</i>	<i>ILS</i>	<i>VNS</i>	<i>SA</i>
20-2-2	1800,01	3,32	2,38	1,19	1,65
20-2-4	1800,10	2,51	1,99	1,17	1,64
20-4-2	6,14	4,33	2,50	1,34	1,79
20-4-4	29,73	5,09	2,93	1,63	1,89
20-8-2	1800,04	8,34	3,46	1,99	1,94
20-8-4	850,18	14,64	5,12	3,60	2,00

Fonte: Elaborada pelo autor (2026).

A TAB. 21 mostra que as meta-heurísticas obtiveram soluções de mesma qualidade que o *Gurobi* em tempos computacionais muito inferiores. Enquanto o *Gurobi* atingiu o limite de 1800 segundos em três instâncias e demandou elevado tempo nas demais, todas as meta-

heurísticas encontraram soluções equivalentes em poucos segundos, evidenciando a elevada eficiência das meta-heurísticas para resolver instâncias de pequeno porte.

6.4 Exemplo de soluções obtidas pelas abordagens propostas

Com o objetivo de ilustrar o funcionamento das meta-heurísticas implementadas, foi selecionada a instância 20-4-4, composta por 20 tarefas, 4 máquinas e 4 recursos. Essa instância foi escolhida por representar um problema de pequeno porte e permitir a visualização completa do sequenciamento das tarefas.

Inicialmente, a FIG. 12 apresenta os dados da instância utilizada nos experimentos.

Figura 12 – Estrutura da instância 20-4-4

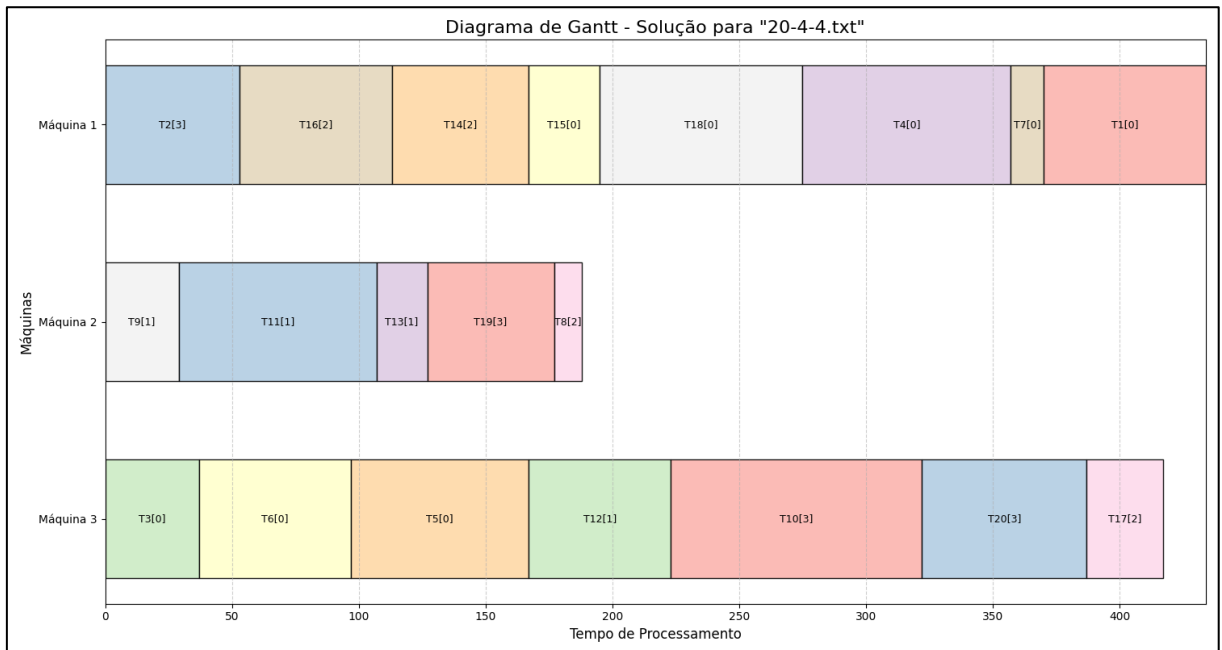
Instância: 20-4-4																			
Número de tarefas (n): 20																			
Número de máquinas (m): 4																			
Número de recursos compartilhados (r): 4																			
Recursos associados às tarefas																			
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
0	3	0	0	0	0	0	2	1	3	1	1	1	2	0	2	2	0	3	3
Tempos de processamento																			
T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T20
64	53	37	82	70	60	13	11	29	99	78	56	20	54	28	60	30	80	50	65

Fonte: Elaborada pelo autor (2026).

Entre os quatro recursos, o recurso 0 representa o principal gargalo. As tarefas associadas a ele somam 434 unidades de tempo e, como não podem ser executadas simultaneamente, qualquer solução precisa de pelo menos esse tempo para ser concluída.

As Figuras 14 a 17 apresentam os gráficos de *Gantt* correspondentes às soluções obtidas pelo *GRASP*, *ILS*, *VNS* e *Simulated Annealing*. Em cada gráfico, as barras representam as tarefas executadas em cada máquina, sendo seu comprimento proporcional ao tempo de processamento. O número entre colchetes identifica o recurso compartilhado associado à tarefa, permitindo verificar que tarefas que utilizam o mesmo recurso não são executadas simultaneamente.

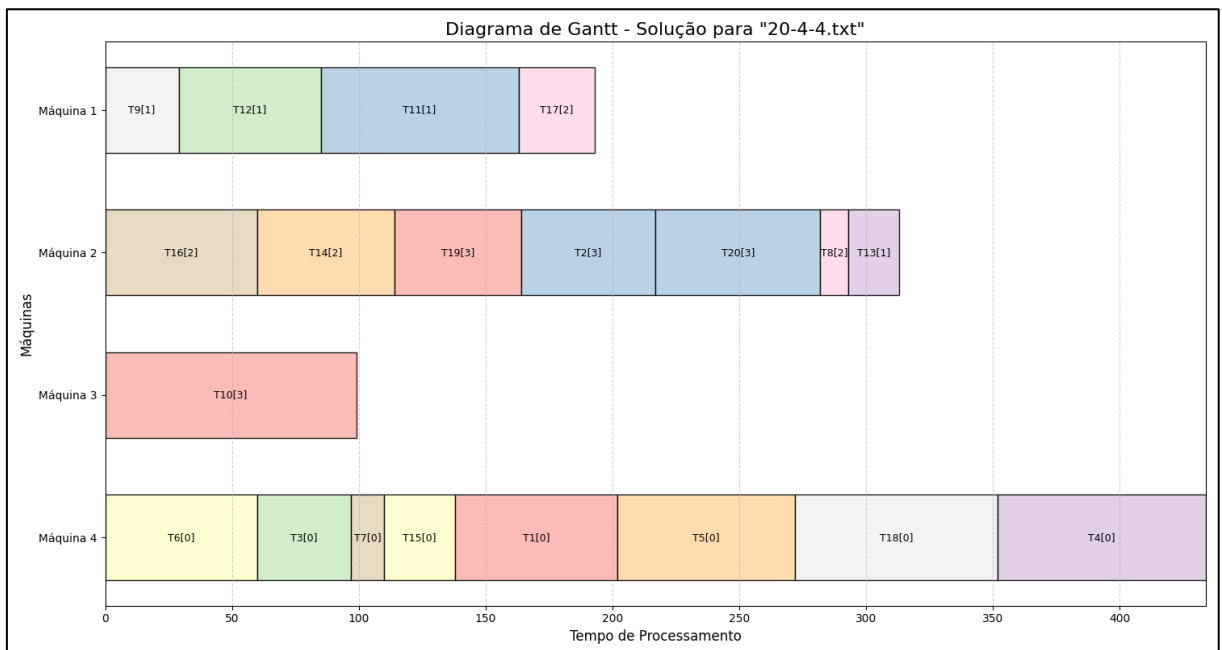
Figura 13 – Gráfico de *Gantt* da solução obtida pelo *GRASP* para a instância 20-4-4.



Fonte: Elaborada pelo autor (2026).

Na solução do *GRASP*, as tarefas foram distribuídas entre três máquinas. Mesmo estando em máquinas diferentes, as tarefas associadas ao recurso 0 foram processadas sem sobreposição. A solução terminou em 434 unidades de tempo.

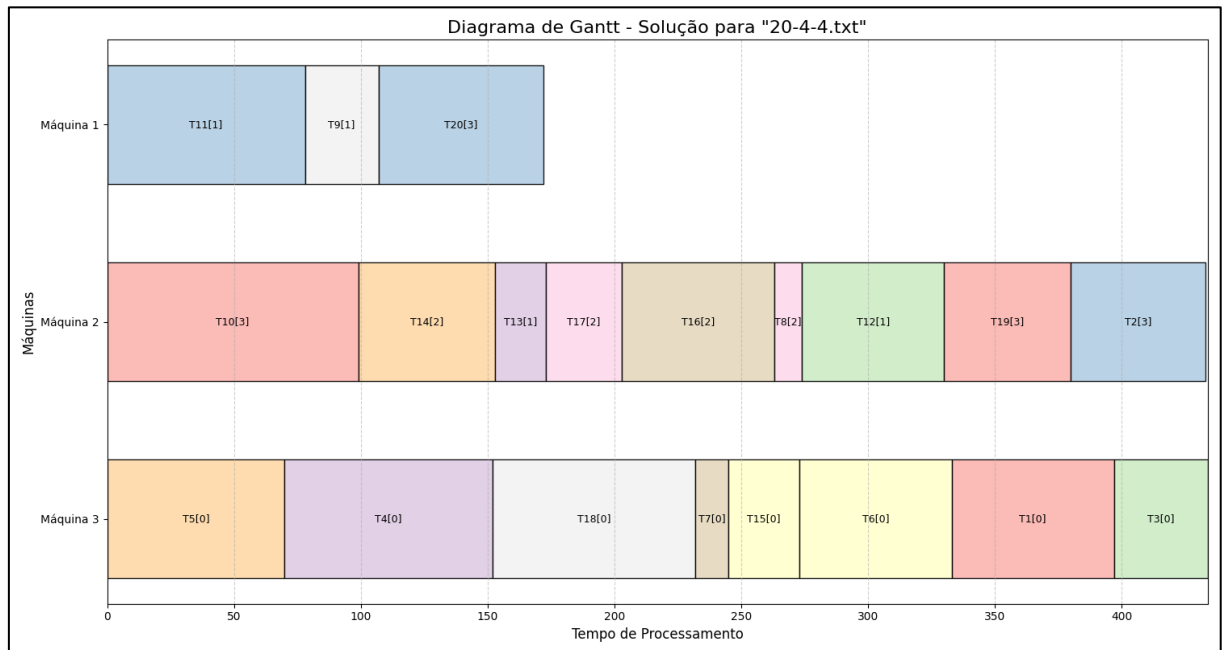
Figura 14 – Gráfico de *Gantt* da solução obtida pelo *ILS* para a instância 20-4-4.



Fonte: Elaborada pelo autor (2026).

O *ILS* utilizou as quatro máquinas disponíveis. As tarefas do recurso 0 foram concentradas na máquina 4, que permaneceu ocupada até o instante 434 e determinou o término da programação.

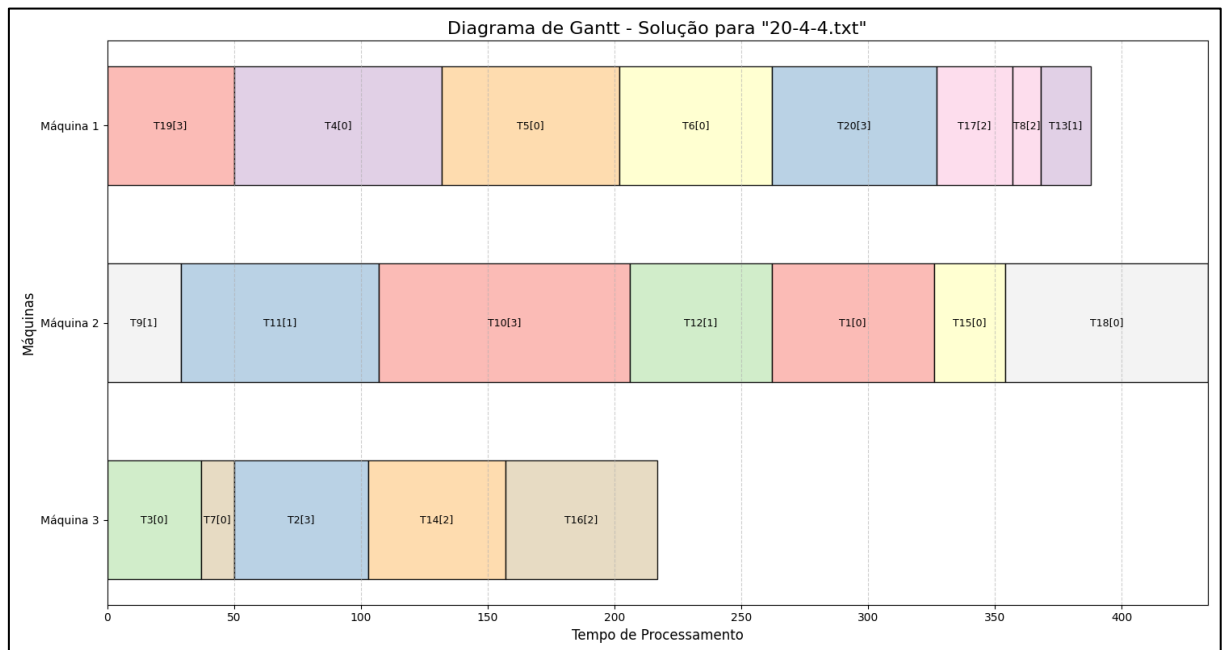
Figura 15 – Gráfico de *Gantt* da solução obtida pelo *VNS* para a instância 20-4-4.



Fonte: Elaborada pelo autor (2026).

Na solução do *VNS*, também foram utilizadas três máquinas. De forma semelhante ao *ILS*, as tarefas associadas ao recurso 0 foram concentradas em uma única máquina e processadas em sequência, sem intervalos de ociosidade. Nesse caso, elas foram alocadas na máquina 3, resultando também em um *makespan* de 434.

Figura 16 – Gráfico de *Gantt* da solução obtida pelo *Simulated Annealing* para a instância 20-4-4.



Fonte: Elaborada pelo autor (2026).

Por fim, o *Simulated Annealing* também utilizou três máquinas. Nesse caso, as tarefas do recurso 0 foram distribuídas entre diferentes máquinas, mas continuaram sendo executadas uma após a outra, respeitando a restrição de compartilhamento. O valor final do *makespan* foi novamente de 434.

Apesar de apresentarem sequenciamentos diferentes, as quatro meta-heurísticas encontraram soluções com o mesmo *makespan*. O *ILS* utilizou todas as máquinas, enquanto *GRASP*, *VNS* e *Simulated Annealing* utilizaram apenas três.

O uso de somente três máquinas não representa um erro. Como o recurso 0 já exige 434 unidades de processamento sequencial, a utilização da quarta máquina não reduziria o tempo total da programação. Ela apenas alteraria a distribuição das tarefas no gráfico.

Assim, todas as soluções respeitaram as restrições do problema e atingiram o limite de 434 unidades de tempo e a análise dos gráficos mostra que, mesmo adotando estratégias de busca distintas, as meta-heurísticas construíram cronogramas eficientes e viáveis. Assim, a representação por meio dos gráficos de *Gantt* facilita a compreensão das soluções obtidas e complementa os resultados quantitativos apresentados nas seções anteriores.

7 CONCLUSÃO

Este trabalho teve como objetivo implementar, calibrar e comparar as meta-heurísticas *Greedy Randomized Adaptive Search Procedure (GRASP)*, *Iterated Local Search (ILS)*, *Variable Neighborhood Search (VNS)* e *Simulated Annealing* para a resolução do problema de máquinas paralelas idênticas com restrição de recursos compartilhados, considerando a minimização do *makespan*.

Após o desenvolvimento dos códigos, foi realizada uma etapa de calibração dos parâmetros por meio de planejamento experimental, permitindo identificar configurações capazes de proporcionar um bom equilíbrio entre qualidade das soluções e tempo computacional. Posteriormente, as meta-heurísticas foram avaliadas utilizando tanto instâncias clássicas da literatura quanto instâncias desenvolvidas especificamente para este trabalho, possibilitando analisar seu comportamento em diferentes configurações do problema.

Os resultados obtidos demonstraram que todas as meta-heurísticas implementadas produziram soluções de elevada qualidade. Em diversas instâncias, o *makespan* coincidiu com o limite inferior, comprovando a otimalidade das soluções, enquanto nos demais casos foram observados baixos desvios em relação a esse limite. Na comparação com o modelo de Programação Linear Inteira Mista resolvido pelo *Gurobi*, verificou-se que as meta-heurísticas encontraram soluções equivalentes para todas as instâncias de pequeno porte analisadas, porém com tempos computacionais significativamente inferiores em diversos casos, evidenciando sua eficiência para esse tipo de problema.

À medida que o porte das instâncias aumentou, tornaram-se mais evidentes as diferenças de desempenho entre os algoritmos. Embora *GRASP*, *ILS* e *VNS* tenham apresentado resultados satisfatórios, o *Simulated Annealing* destacou-se de forma consistente, obtendo, na maioria das instâncias de médio e grande porte, os menores desvios percentuais em relação ao limite inferior, os menores tempos computacionais e a menor variabilidade entre as execuções. Esses resultados indicam que o SA apresentou melhor equilíbrio entre qualidade das soluções, tempo de processamento e estabilidade nos problemas de maior complexidade.

Além da comparação entre as meta-heurísticas, este trabalho também contribuiu com a geração de um novo conjunto de instâncias para o problema de máquinas paralelas com restrição de recursos, contemplando diferentes combinações entre número de tarefas, máquinas e recursos compartilhados. Essas instâncias ampliam as possibilidades de avaliação de métodos de solução

para um problema mais geral do que aquele tradicionalmente encontrado na literatura, normalmente restrito ao caso de duas máquinas paralelas.

Como principal contribuição, destaca-se a implementação de quatro meta-heurísticas clássicas aplicadas a uma mesma formulação do problema, a realização de uma calibração sistemática de seus parâmetros, a comparação experimental entre os métodos e a adaptação de um modelo matemático da literatura para um número arbitrário de máquinas paralelas. Os resultados obtidos demonstram que as meta-heurísticas constituem uma alternativa eficiente para problemas de sequenciamento com restrições de recursos compartilhados, produzindo soluções de elevada qualidade em tempos compatíveis com aplicações práticas de apoio à tomada de decisão.

Como desdobramento desta pesquisa, uma versão parcial dos resultados foi submetida ao Encontro Nacional de Engenharia de Produção (ENEGEP), sob o título “*GRASP*, *ILS* e *VNS* aplicados ao problema de máquinas paralelas com restrição de recursos”, sendo aprovada para apresentação no evento. O trabalho submetido contemplou apenas os resultados das meta-heurísticas *GRASP*, *ILS* e *VNS*, uma vez que os experimentos com o *Simulated Annealing* ainda não haviam sido realizados naquele momento.

Como limitações deste estudo, destaca-se que a comparação com o modelo matemático foi restrita às instâncias de pequeno porte, em função da elevada complexidade computacional do problema. Além disso, foram consideradas apenas máquinas paralelas idênticas, sem a inclusão de características presentes em ambientes industriais reais, como tempos de preparação dependentes da sequência, datas de entrega, indisponibilidade de máquinas e múltiplos recursos compartilhados por tarefa.

Como trabalhos futuros, sugere-se a investigação de outras meta-heurísticas, bem como o desenvolvimento de abordagens híbridas que combinem diferentes estratégias de busca. Também recomenda-se a utilização de técnicas automáticas de ajuste de parâmetros, como o *irace*, visando obter configurações mais eficientes para os algoritmos. Além disso, propõe-se a aplicação dos métodos desenvolvidos em instâncias reais de ambientes industriais, permitindo avaliar seu desempenho em problemas práticos e identificar oportunidades de adaptação às características específicas dos sistemas produtivos.

REFERÊNCIAS BIBLIOGRÁFICAS

ARENALES, Marcos et al. **Pesquisa Operacional para Cursos de Engenharia**. Rio de Janeiro: Elsevier, 2007.

BAKER, Kenneth R.; TRIETSCH, Dan. *Principles of Sequencing and Scheduling*. 2. ed. Hoboken: John Wiley & Sons, 2019.

ČERNÝ, V. **Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm**. *Journal of Optimization Theory and Applications*, v. 45, n. 1, p. 41–51, jan. 1985.

CHUNG, Tsuiping; GUPTA, Jatinder N. D.; ZHAO, Haidan; WERNER, Frank. **Minimizing the makespan on two identical parallel machines with mold constraints**. *Computers & Operations Research*, v. 105, p. 141–155, 2019. DOI: 10.1016/j.cor.2019.01.005.

CONWAY, Richard W.; MAXWELL, William L.; MILLER, Louis W. *Theory of Scheduling*. Reading: Addison-Wesley, 1967.

CORRÊA, Luciano Alves. **Meta-heurísticas para redução do makespan em um problema de sequenciamento de tarefas em duas máquinas paralelas e idênticas com restrição de recursos**. 2025. 85 f. Dissertação (Mestrado Acadêmico em Engenharia de Produção) – Departamento de Engenharia de Produção, Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Ouro Preto, Ouro Preto, 2025.

FEO, Thomas A.; RESENDE, Mauricio G. C. **Greedy randomized adaptive search procedures**. *Journal of Global Optimization*, v. 6, n. 2, p. 109–133, 1995. DOI: 10.1007/BF01096763.

GAREY, Michael R.; JOHNSON, David S.; SETHI, Ravi. **The complexity of flowshop and jobshop scheduling**. *Mathematics of Operations Research*, v. 1, n. 2, p. 117–129, 1976. DOI: 10.1287/moor.1.2.117.

GONÇALVES, José Fernando; MENDES, Jorge José de Magalhães; RESENDE, Maurício G. C. **A hybrid genetic algorithm for the job shop scheduling problem**. Florham Park: AT&T Labs Research, 2002.

GUROBI OPTIMIZATION, LLC. **Gurobi Optimizer Reference Manual**. Beaverton, OR, USA, 2025. Disponível em: <https://www.gurobi.com/documentation/>. Acesso em: 22 jun. 2026.

HILLIER, Frederick S.; LIEBERMAN, Gerald J. **Introduction to Operations Research**. 10. ed. New York: McGraw-Hill, 2013.

KAABI, A.; HARRATH, Y. **Survey of parallel machine scheduling problems and heuristics**. *International Journal of Computer Applications*, v. 90, n. 2, p. 1–8, 2014.

KIRKPATRICK, Scott; GELATT, C. Daniel; VECCHI, Mario P. **Optimization by simulated annealing**. *Science*, v. 220, n. 4598, p. 671–680, 1983.

LAWLER, Eugene L.; LENSTRA, Jan Karel; RINNOOY KAN, Alexander H. G.; SHMOYS, David B. **Sequencing and scheduling: algorithms and complexity**. In: GRAVES, Stephen C.; RINNOOY KAN, Alexander H. G.; ZIPKIN, Paul H. (ed.). *Handbooks in Operations Research and Management Science*. Amsterdam: Elsevier, 1989. v. 4, p. 445–522.

LOPES, Heitor Silvério; RODRIGUES, Luiz Carlos de Abreu; STEINER, Maria Teresinha Arns. **Meta-Heurísticas em Pesquisa Operacional**. Curitiba: Omnipax, 2013.

LOURENÇO, Helena R.; MARTIN, Olivier C.; STÜTZLE, Thomas. **Iterated Local Search**. In: GLOVER, Fred; KOCHENBERGER, Gary A. (ed.). *Handbook of Metaheuristics*. Boston: Springer, 2003. p. 320–353. DOI: 10.1007/0-306-48056-5_11.

MLADENović, Nenad; HANSEN, Pierre. **Variable Neighborhood Search**. *Computers & Operations Research*, v. 24, n. 11, p. 1097–1100, 1997.

MONTGOMERY, Douglas C.; RUNGER, George C. **Applied statistics and probability for engineers**. 3. ed. New York: John Wiley & Sons, 2003.

MÜLLER, Felipe Martins; DIAS, Odon Bastos; ARAÚJO, Olinto César Bassi de. **Algoritmo para o problema de sequenciamento em máquinas paralelas não-relacionadas**. *Produção*, v. 12, n. 2, p. 6–17, 2002.

PINEDO, Michael L. **Scheduling: Theory, Algorithms, and Systems**. 5. ed. New York: Springer, 2016.

RESENDE, Mauricio G. C.; RIBEIRO, Celso C. **Optimization by GRASP: Greedy Randomized Adaptive Search Procedures**. New York: Springer, 2016. DOI: 10.1007/978-1-4939-6530-4.

SAYADI, Mohammad Kazem; RAMEZANIAN, Reza; GHAFARI-NASAB, Nader. **A discrete firefly meta-heuristic with local search for makespan minimization in**

permutation flow shop scheduling problems. *International Journal of Industrial Engineering Computations*, v. 1, n. 1, p. 1–10, 2010. DOI: 10.5267/j.ijiec.2010.01.001.

SLACK, Nigel; BRANDON-JONES, Alistair; BURGESS, Nicola. *Administração da Produção*. 10. ed. Rio de Janeiro: Atlas, 2023.

SOARES, L. C. R.; CARVALHO, M. A. M. **Biased random-key genetic algorithm for identical parallel machine scheduling with tooling constraints.** *European Journal of Operational Research*, v. 285, n. 3, p. 955–964, 2020. DOI: 10.1016/j.ejor.2020.02.047.

SOARES, Rodrigo G.; CARVALHO, Claudio C. **Identical Parallel Machine Scheduling Problem with Tooling Constraints and Sequence-Dependent Setup Times.** *Expert Systems with Applications*, v. 202, 2022. DOI: 10.1016/j.eswa.2022.117338.

SOUZA, Marcone Jamilson Freitas. **Inteligência computacional para otimização.** Ouro Preto: Universidade Federal de Ouro Preto, Departamento de Computação, 2 fev. 2024. Notas de aula. Disponível em:

<https://www.decom.ufop.br/prof/marcone/Disciplinas/InteligenciaComputacional/InteligenciaComputacional.pdf>. Acesso em: 25 jun. 2026.

TALBI, El-Ghazali. *Metaheuristics: From Design to Implementation*. Hoboken: John Wiley & Sons, 2009.

TUBINO, Dalvio Ferrari. *Planejamento e Controle da Produção: Teoria e Prática*. 3. ed. Rio de Janeiro: Atlas, 2017.

VENTURA, José A.; KIM, Donghyun. **Parallel machine scheduling with earliness–tardiness penalties and additional resource constraints.** *Computers & Operations Research*, v. 30, n. 13, p. 1945–1958, 2003. DOI: 10.1016/S0305-0548(02)00118-1.

WOLSEY, Laurence A. *Integer Programming*. New York: Wiley, 1998.

ZHANG, Y.; CHEN, Z.; LI, J. **Approximation algorithms for parallel machine scheduling with renewable resource constraints.** *Theoretical Computer Science*, v. 974, 2023. DOI: 10.1016/j.tcs.2023.114126.

APÊNDICE

APÊNDICE A: RESULTADOS COMPLETOS DAS INSTÂNCIAS DE CHUNG ET AL. (2019)

APÊNDICE A.1: Resultados completos dos testes das instâncias de Chung et al. (2019) com 20 e 50 tarefas

Fonte: Autor (2026)

instancia	j	m	r	heuristica	lowerbound	media_mak	var_mak	dp_mak	media_gap	var_gap	dp_gap	media_tempo	var_tempo	dp_tempo	melhor_mak	melhor_tempo	seed_melhor_mak
HA001.txt	20	2	4	GRASP	593.5	594	0	0.0	0.5	0.0	0.0	2.555737741788	0.0285890726	0.16908303464	594	2.721225976943969	1776974686
HA001.txt	20	2	4	ILS	593.5	594	0	0.0	0.5	0.0	0.0	1.890973607699	0.0839744877	0.28978351884	594	1.552497863769531	1776721476
HA001.txt	20	2	4	VNS	593.5	594	0	0.0	0.5	0.0	0.0	1.169116163253	0.0314611340	0.17737286731	594	0.969883918762207	1776721542
HA001.txt	20	2	4	SA	593.5	594	0	0.0	0.5	0.0	0.0	1.610541	0.055786	0.236191	594	1.700287	1780779722
HA001.txt	20	2	6	GRASP	435.0	435	0	0.0	0.0	0.0	0.0	1.378699525197	0.0096907941	0.09844183138	435	1.442367076873779	1776721601
HA001.txt	20	2	6	ILS	435.0	435	0	0.0	0.0	0.0	0.0	1.373504130045	0.0137057973	0.11707176152	435	1.426760673522949	1776721652
HA001.txt	20	2	6	VNS	435.0	435	0	0.0	0.0	0.0	0.0	1.005315446853	0.0102809305	0.10139492377	435	0.987238407135009	1776721709
HA001.txt	20	2	6	SA	435.0	435	0	0.0	0.0	0.0	0.0	1.604824	0.005313	0.072894	435	1.5771	1780779779
HA001.txt	20	2	8	GRASP	490.0	490	0	0.0	0.0	0.0	0.0	1.175269563992	0.0064908009	0.08056550748	490	1.293319940567016	1776721773
HA001.txt	20	2	8	ILS	490.0	490	0	0.0	0.0	0.0	0.0	1.337574831644	0.0184494504	0.13582875429	490	1.167823553085327	1776721812
HA001.txt	20	2	8	VNS	490.0	490	0	0.0	0.0	0.0	0.0	1.026350990931	0.0107034113	0.10345729235	490	1.145464658737182	1776721862
HA001.txt	20	2	8	SA	490.0	490	0	0.0	0.0	0.0	0.0	1.569298	0.048195	0.219533	490	1.728276	1780779834
HA001.txt	20	2	10	GRASP	537.0	537	0	0.0	0.0	0.0	0.0	0.881548237800	0.0015924852	0.03990595504	537	0.934980154037479	1776721939
HA001.txt	20	2	10	ILS	537.0	537	0	0.0	0.0	0.0	0.0	1.111138168970	0.0057740296	0.07598703609	537	1.088093042373657	1776721969
HA001.txt	20	2	10	VNS	537.0	537	0	0.0	0.0	0.0	0.0	0.873249403635	0.0034081393	0.05837927158	537	0.889478683471679	1776722006
HA001.txt	20	2	10	SA	537.0	537	0	0.0	0.0	0.0	0.0	1.598179	0.038124	0.195255	537	1.702106	1780779887
HA001.txt	50	2	4	GRASP	1171.0	1171	0	0.0	0.0	0.0	0.0	316.7042962630	34.927237470	5.90992702747	1171	314.3506407737732	1776731666
HA001.txt	50	2	4	ILS	1171.0	1171	0	0.0	0.0	0.0	0.0	234.8649441878	1883.1137166	43.3948581826	1171	243.5024030208587	1776766221
HA001.txt	50	2	4	VNS	1171.0	1171	0	0.0	0.0	0.0	0.0	104.5989716529	1029.2403649	32.0817762132	1171	93.44639158248901	1776780095
HA001.txt	50	2	4	SA	1171.0	1171	0	0.0	0.0	0.0	0.0	9.391479	1.400685	1.183506	1171	9.418056	1780779951
HA001.txt	50	2	6	GRASP	1092.5	1093	0	0.0	0.5	0.0	0.0	182.3300498247	50.856945790	7.13140559713	1093	190.2107992172241	1776788113
HA001.txt	50	2	6	ILS	1092.5	1093	0	0.0	0.5	0.0	0.0	113.1184698979	762.68592066	27.6167688310	1093	68.36797213554382	1776793734
HA001.txt	50	2	6	VNS	1092.5	1093	0	0.0	0.5	0.0	0.0	55.32914148966	97.196678522	9.85883758476	1093	54.55458211898804	1776797177
HA001.txt	50	2	6	SA	1092.5	1093	0	0.0	0.5	0.0	0.0	8.749004	1.737242	1.318045	1093	10.37048	1780780240
HA001.txt	50	2	8	GRASP	1109.0	1109	0	0.0	0.0	0.0	0.0	108.7159293095	28.800087268	5.36657127673	1109	111.695007562673	1776799530
HA001.txt	50	2	8	ILS	1109.0	1109	0	0.0	0.0	0.0	0.0	71.24247719446	243.36113909	15.6000365094	1109	84.5533378124237	1776803057
HA001.txt	50	2	8	VNS	1109.0	1109	0	0.0	0.0	0.0	0.0	38.61904065608	32.869722445	5.73321222750	1109	38.72324299812317	1776805204
HA001.txt	50	2	8	SA	1109.0	1109	0	0.0	0.0	0.0	0.0	9.369635	0.305259	0.552503	1109	9.984473	1780780515
HA001.txt	50	2	10	GRASP	1233.0	1233	0	0.0	0.0	0.0	0.0	71.34168372948	9.0855546613	3.01422538329	1233	73.72340154647827	1776811673
HA001.txt	50	2	10	ILS	1233.0	1233	0	0.0	0.0	0.0	0.0	51.57125777403	52.224647921	7.22666229469	1233	45.47063493728638	1776814717
HA001.txt	50	2	10	VNS	1233.0	1233	0	0.0	0.0	0.0	0.0	34.24313289324	12.580436681	3.54689112904	1233	37.64152240753174	1776806441
HA001.txt	50	2	10	SA	1233.0	1233	0	0.0	0.0	0.0	0.0	9.18012	0.614515	0.78391	1233	9.192298	1780780819

APÊNDICE A.2: Resultados completos dos testes das instâncias de Chung et al. (2019) com 100 e 150 tarefas

Fonte: Autor (2026)

instancia	j	m	r	heuristica	lowerbound	media_mak	var_mak	dp_mak	media_gap	var_gap	dp_gap	media_tempo	var_tempo	dp_tempo	melhor_mak	melhor_tempo	seed_melhor_mak
HA001.txt	100	2	4	GRASP	2601.5	2602	0	0.0	0.5	0.0	0.0	326.6149355729	247.80327920	15.7417686174	2602	343.9838533401489	1776939814
HA001.txt	100	2	4	ILS	2601.5	2602	0	0.0	0.5	0.0	0.0	323.8721083799	191.99584318	13.8562564634	2602	325.6350569725036	1776910972
HA001.txt	100	2	4	VNS	2601.5	2602	0	0.0	0.5	0.0	0.0	321.1316167354	193.29904727	13.9032027703	2602	300.1067476272583	1776964734
HA001.txt	100	2	4	SA	2601.5	2602	0	0.0	0.5	0.0	0.0	36.885323	27.647916	5.258129	2602	39.57525	1780781107
HA001.txt	100	2	6	GRASP	2552.5	2553	0	0.0	0.5	0.0	0.0	321.3654452641	182.27710184	13.5010037348	2553	300.3872950077057	1776974904
HA001.txt	100	2	6	ILS	2552.5	2553	0	0.0	0.5	0.0	0.0	323.4670179764	169.64878205	13.0249292533	2553	304.8732526302337	1776984654
HA001.txt	100	2	6	VNS	2552.5	2553	0	0.0	0.5	0.0	0.0	316.6367074092	139.01570885	11.7904923076	2553	300.8731868267059	1776994377
HA001.txt	100	2	6	SA	2552.5	2553	0	0.0	0.0	0.0	0.0	38.00518	13.212668	3.634923	2553	40.358059	1780782253
HA001.txt	100	2	8	GRASP	2357.0	2357	0	0.0	0.0	0.0	0.0	314.6384333769	99.947817556	9.99739053736	2357	301.9350965023041	1777026119
HA001.txt	100	2	8	ILS	2357.0	2357	0	0.0	0.0	0.0	0.0	318.2510131676	123.34566557	11.1061093803	2357	328.632342338562	1777035565
HA001.txt	100	2	8	VNS	2357.0	2357	0	0.0	0.0	0.0	0.0	313.2683567206	134.41314775	11.5936684337	2357	311.0756397247314	1777045286
HA001.txt	100	2	8	SA	2357.0	2357	0	0.0	0.0	0.0	0.0	36.884301	16.292802	4.036434	2357	39.759049	1780783410
HA001.txt	100	2	10	GRASP	2521.0	2521	0	0.0	0.0	0.0	0.0	315.3217447678	172.49236920	13.1336350340	2521	302.1651887893677	1777054738
HA001.txt	100	2	10	ILS	2521.0	2521	0	0.0	0.0	0.0	0.0	311.5076212008	77.382719560	8.79674482750	2521	304.1965570449829	1777064495
HA001.txt	100	2	10	VNS	2521.0	2521	0	0.0	0.0	0.0	0.0	307.1813993533	89.588160174	9.46510222737	2521	304.0643813610077	1777073842
HA001.txt	100	2	10	SA	2521.0	2521	0	0.0	0.0	0.0	0.0	35.796608	33.111207	5.754234	2521	37.912712	1780784574
HA001.txt	150	2	4	GRASP	3715.5	3716.7	5.8724137	2.4233063	1.2	5.8724137	2.4233063	719.5607006311	1759.0251082	41.9407332827	3716	722.0576636791229	1777107506
HA001.txt	150	2	4	ILS	3715.5	3717.4	17.696551	4.2067269	1.9	17.696551	4.2067269	707.4267346779	4712.2415971	68.6457689673	3716	623.5251996517181	1777129118
HA001.txt	150	2	4	VNS	3715.5	3716.1	0.0931034	0.3051285	0.6	0.0931034	0.3051285	725.1484818696	4386.9963583	66.2344046424	3716	673.5001122951508	1777150387
HA001.txt	150	2	4	SA	3715.5	3716	0	0.0	0.5	0.0	0.0	83.302068	132.372502	11.505325	3716	74.890249	1780785845
HA001.txt	150	2	6	GRASP	3694.0	3694	0	0.0	0.0	0.0	0.0	679.2996487061	3944.5953095	62.8060133234	3694	662.7841818332672	1777302582
HA001.txt	150	2	6	ILS	3694.0	3694	0	0.0	0.0	0.0	0.0	709.1482134103	4780.4369242	69.1407038166	3694	628.0770087242126	1777322993
HA001.txt	150	2	6	VNS	3694.0	3694	0.0643678	0.2537081	0.0666666666	0.0643678	0.2537081	686.9559146563	2266.1733724	47.6043419500	3694	722.0516481399536	1777344287
HA001.txt	150	2	6	SA	3694.0	3694	0	0.0	0.000000	0.000000	0.000000	83.14978	157.45701	12.548188	3694	88.398667	1780788399
HA001.txt	150	2	8	GRASP	3630.5	3631	0	0.0	0.5	0.0	0.0	704.2629047473	2179.1846922	46.6817383169	3631	699.906174659729	1777372872
HA001.txt	150	2	8	ILS	3630.5	3631	0	0.0	0.5	0.0	0.0	691.4694471279	2461.4081705	49.6125807688	3631	768.7578392028809	1777394436
HA001.txt	150	2	8	VNS	3630.5	3631	0	0.0	0.5	0.0	0.0	693.7612978617	3663.2984418	60.5251884906	3631	759.623398065567	1777417977
HA001.txt	150	2	8	SA	3630.5	3631	0	0.0	0.5	0.0	0.0	83.278894	122.991157	11.090138	3631	82.242815	1780793766
HA001.txt	150	2	10	GRASP	4270.0	4270	0	0.0	0.0	0.0	0.0	685.8472800652	3732.8728110	61.0972406170	4270	694.1695425510406	1777478375
HA001.txt	150	2	10	ILS	4270.0	4270	0	0.0	0.0	0.0	0.0	688.5963550488	3158.7820373	56.2030429543	4270	788.3283898830414	1777509880
HA001.txt	150	2	10	VNS	4270.0	4270	0	0.0	0.0	0.0	0.0	670.7083292881	4263.6739141	65.2968139658	4270	668.0506010055542	1777458186
HA001.txt	150	2	10	SA	4270.0	4270	0	0.0	0.000000	0.000000	0.000000	84.496893	125.420499	11.199129	4270	92.158172	1780796280

APÊNDICE B: RESULTADOS COMPLETOS DAS INSTÂNCIAS GERADAS PARA O TRABALHO

APÊNDICE B.1: Resultados completos dos testes das instâncias geradas para o trabalho com 20 tarefas

Fonte: Autor (2026)

INSTANCIA	HEURISTICA	LB	J	M	R	MED_MAK	VARIANCIA_MAK	DVP_MAK	MEDIA_GAP	VAR_GAP	DVP_GAP	MEDIA_TEMPO	VAR_TEMPO	DVP_TEMPO	MELHOR_MAK	MELHOR_TEMPO	SEED_MELHOR_MAK
20-2-2.txt	GRASP	598	20	2	2	627	0	0	29	0	0	3.32	0.11	0.33	627	2.82	1781051661
20-2-2.txt	ILS	598	20	2	2	627	0	0	29	0	0	2.38	0.19	0.44	627	2.08	1781051768
20-2-2.txt	VNS	598	20	2	2	627	0	0	29	0	0	1.19	0.05	0.23	627	0.98	1781051849
20-2-2.txt	SA	598	20	2	2	627	0	0	29	0	0	1.65	0.04	0.21	627	1.85	1781052012
20-2-4.txt	GRASP	572.5000	20	2	4	573	0	0	0.5	0	0	2.51	0.04	0.19	573	2.44	1781052089
20-2-4.txt	ILS	572.5000	20	2	4	573	0	0	0.5	0	0	1.99	0.11	0.33	573	2.79	1781052169
20-2-4.txt	VNS	572.5000	20	2	4	573	0	0	0.5	0	0	1.17	0.03	0.19	573	1.07	1781052236
20-2-4.txt	SA	572.5000	20	2	4	573	0	0	0.5	0	0	1.64	0.04	0.20	573	1.63	1781052279
20-4-2.txt	GRASP	208	20	4	2	429	0	0	221	0	0	4.33	0.10	0.32	429	4.77	1781052438
20-4-2.txt	ILS	208	20	4	2	429	0	0	221	0	0	2.50	0.13	0.35	429	2.97	1781052576
20-4-2.txt	VNS	208	20	4	2	429	0	0	221	0	0	1.34	0.09	0.30	429	1.06	1781052658
20-4-2.txt	SA	208	20	4	2	429	0	0	221	0	0	1.79	0.01	0.08	429	1.72	1781052707
20-4-4.txt	GRASP	230.5000	20	4	4	434	0	0	203.5	0	0	5.09	0.09	0.31	434	5.24	1781052791
20-4-4.txt	ILS	230.5000	20	4	4	434	0	0	203.5	0	0	2.93	0.21	0.45	434	2.39	1781052947
20-4-4.txt	VNS	230.5000	20	4	4	434	0	0	203.5	0	0	1.63	0.06	0.24	434	1.49	1781053041
20-4-4.txt	SA	230.5000	20	4	4	434.1	0.16	0.40	203.6	0.16	0.4026	1.89	0.00	0.05	434	1.87	1781053095
20-8-2.txt	GRASP	101.1250	20	8	2	414	0	0	312.88	0	0	8.34	0.20	0.45	414	8.13	1781053155
20-8-2.txt	ILS	101.1250	20	8	2	414	0	0	312.88	0	0	3.46	0.91	0.95	414	3.52	1781053408
20-8-2.txt	VNS	101.1250	20	8	2	414	0	0	312.88	0	0	1.99	0.12	0.34	414	2.31	1781053609
20-8-2.txt	SA	101.1250	20	8	2	414	0	0	312.88	0	0	1.94	0.01	0.08	414	2.13	1781053676
20-8-4.txt	GRASP	110.1250	20	8	4	316.6	34.32	5.86	206.48	34.3172	5.8581	14.64	34.94	5.91	314	11.09	1781053770
20-8-4.txt	ILS	110.1250	20	8	4	314	0	0	203.88	0	0	5.12	2.89	1.70	314	2.89	1781054217
20-8-4.txt	VNS	110.1250	20	8	4	314	0	0	203.88	0	0	3.60	0.83	0.91	314	2.42	1781054374
20-8-4.txt	SA	110.1250	20	8	4	318.13	76.33	8.74	208.01	76.3264	8.7365	2.00	0.01	0.10	314	1.91	1781054487

APÊNDICE B.2: Resultados completos dos testes das instâncias geradas para o trabalho com 50 tarefas

Fonte: Autor (2026)

INSTANCIA	HEURISTICA	LB	J	M	R	MED_MAK	VARIANCIA_MAK	DVP_MAK	MEDIA_GAP	VAR_GAP	DVP_GAP	MEDIA_TEMPO	VAR_TEMPO	DVP_TEMPO	MELHOR_MAK	MELHOR_TEMPO	SEED_MELHOR_MAK
50-2-2.txt	GRASP	1355.0000	50	2	2	1402	0	0	47	0	0	270.77	162.60	12.75	1402	276.47	1781054567
50-2-2.txt	ILS	1355.0000	50	2	2	1402	0	0	47	0	0	200.11	3178.74	56.38	1402	256.02	1781062729
50-2-2.txt	VNS	1355.0000	50	2	2	1402	0	0	47	0	0	73.04	726.40	26.95	1402	56.42	1781088227
50-2-2.txt	SA	1355.0000	50	2	2	1402	0	0	47	0	0	9.45	1.21	1.10	1402	10.49	1781090830
50-2-4.txt	GRASP	1243.5000	50	2	4	1244	0	0	0.5	0	0	363.49	49.76	7.05	1244	358.02	1781091120
50-2-4.txt	ILS	1243.5000	50	2	4	1244	0	0	0.5	0	0	286.96	1897.34	43.56	1244	123.64	1781109863
50-2-4.txt	VNS	1243.5000	50	2	4	1244	0	0	0.5	0	0	128.19	890.70	29.84	1244	139.13	1781118754
50-2-4.txt	SA	1243.5000	50	2	4	1244	0	0	0.5	0	0	8.82	2.10	1.45	1244	9.78	1781122641
50-4-2.txt	GRASP	663.2500	50	4	2	1506	0	0.0000	842.7500	0.0000	0.0000	337.3268	100.6833	10.0341	1506	352.7895	1782580283
50-4-2.txt	ILS	663.2500	50	4	2	1506	0	0.0000	842.7500	0.0000	0.0000	169.1992	1885.3305	43.4204	1506	156.4849	1782590705
50-4-2.txt	VNS	663.2500	50	4	2	1506	0	0.0000	842.7500	0.0000	0.0000	70.3825	1041.7222	32.2757	1506	129.6914	1782596294
50-4-2.txt	SA	663.2500	50	4	2	1506	0	0.0000	842.7500	0.0000	0.0000	10.2733	0.0504	0.2246	1506	9.9523	1782598426
50-4-4.txt	GRASP	697.2500	50	4	4	946.17	1110.21	33.32	248.92	1110.21	33.32	859.62	78719.58	280.57	873	907.94	1781128468
50-4-4.txt	ILS	697.2500	50	4	4	884.57	1199.98	34.64	187.32	1199.98	34.64	608.16	40485.83	201.21	836	963.54	1781226472
50-4-4.txt	VNS	697.2500	50	4	4	885.53	1064.40	32.63	188.28	1064.40	32.63	319.26	12354.26	111.15	836	177.65	1781259498
50-4-4.txt	SA	697.2500	50	4	4	874.2	639.61	25.29	176.95	639.61	25.29	10.68	0.06	0.25	836	10.71	1781269329
50-8-2.txt	GRASP	333.7500	50	8	2	1400	0	0	1066.25	0	0	315.53	246.57	15.70	1400	318.17	1781269672
50-8-2.txt	ILS	333.7500	50	8	2	1400	0	0	1066.25	0	0	193.31	7706.20	87.78	1400	167.18	1781295230
50-8-2.txt	VNS	333.7500	50	8	2	1400	0	0	1066.25	0	0	78.97	1204.43	34.70	1400	54.40	1781301363
50-8-2.txt	SA	333.7500	50	8	2	1400	0	0	1066.25	0	0	11.16	0.15	0.39	1400	11.60	1781303746
50-8-4.txt	GRASP	296.8750	50	8	4	785.1	317.82	17.83	488.23	317.82	17.83	1010.31	94624.30	307.61	751	1249.76	1781304109
50-8-4.txt	ILS	296.8750	50	8	4	740.9	564.23	23.75	444.03	564.23	23.75	670.18	59606.19	244.14	706	513.36	1781349131
50-8-4.txt	VNS	296.8750	50	8	4	745.17	538.14	23.20	448.29	538.14	23.20	381.62	20505.96	143.20	706	167.52	1781370570
50-8-4.txt	SA	296.8750	50	8	4	738.9	413.33	20.33	442.03	413.33	20.33	11.48	0.11	0.34	706	10.93	1781382162

APÊNDICE B.3: Resultados completos dos testes das instâncias geradas para o trabalho com 100 tarefas

Fonte: Autor (2026)

INSTANCIA	HEURISTICA	LB	J	M	R	MED_MAK	VARIANCIA_MAK	DVP_MAK	MEDIA_GAP	VAR_GAP	DVP_GAP	MEDIA_TEMPO	VAR_TEMPO	DVP_TEMPO	MELHOR_MAK	MELHOR_TEMPO	SEED_MELHOR_MAK
100-2-2.txt	GRASP	2260.0000	100	2	2	2751.3	305.87	17.49	491.3	305.87	17.49	320.90	314.81	17.74	2747	318.49	1781382560
100-2-2.txt	ILS	2260.0000	100	2	2	2747	0	0	487	0	0	320.51	193.75	13.92	2747	341.55	1781427821
100-2-2.txt	VNS	2260.0000	100	2	2	2747	0	0	487	0	0	316.13	203.26	14.26	2747	328.17	1781474047
100-2-2.txt	SA	2260.0000	100	2	2	2747	0	0	487	0	0	40.07	4.82	2.19	2747	35.26	1781523396
100-2-4.txt	GRASP	2740.0000	100	2	4	2740.03	0.0333	0.18	0.03	0.03	0.18	326.50	195.43	13.98	2740	324.20	1781524617
100-2-4.txt	ILS	2740.0000	100	2	4	2740.1	0.0931	0.31	0.1	0.09	0.31	324.38	210.71	14.52	2740	303.76	1781541869
100-2-4.txt	VNS	2740.0000	100	2	4	2740	0	0	0	0	0	318.59	184.55	13.58	2740	330.61	1781551795
100-2-4.txt	SA	2740.0000	100	2	4	2740	0	0	0	0	0	38.80	4.68	2.16	2740	39.62	1781570715
100-4-2.txt	GRASP	1211.2500	100	4	2	2492.27	11248.41	106.06	1281.02	11248.41	106.06	313.49	125.69	11.21	2433	302.57	1781571987
100-4-2.txt	ILS	1211.2500	100	4	2	2506.7	33722.84	183.64	1295.45	33722.84	183.64	318.53	192.60	13.88	2433	304.20	1781581420
100-4-2.txt	VNS	1211.2500	100	4	2	2517.83	57683.04	240.17	1306.58	57683.04	240.17	314.05	279.66	16.72	2433	317.13	1781606129
100-4-2.txt	SA	1211.2500	100	4	2	2433	0	0	1221.75	0	0	41.81	0.73	0.85	2433	41.42	1781622625
100-4-4.txt	GRASP	1201.7500	100	4	4	1966.57	9276.74	96.32	764.82	9276.74	96.32	334.94	363.43	19.06	1732	338.41	1781624238
100-4-4.txt	ILS	1201.7500	100	4	4	1900.3	16432.36	128.19	698.55	16432.36	128.19	324.47	279.55	16.72	1643	304.41	1781634328
100-4-4.txt	VNS	1201.7500	100	4	4	1874.9	11545.20	107.45	673.15	11545.20	107.45	315.73	213.33	14.61	1693	322.62	1781644198
100-4-4.txt	SA	1201.7500	100	4	4	1545.6	1306.52	36.15	343.85	1306.52	36.15	43.28	1.40	1.18	1394	42.15	1781654719
100-8-2.txt	GRASP	627.7500	100	8	2	2693.6	46532.11	215.71	2065.85	46532.11	215.71	313.87	197.40	14.05	2568	307.22	1781693452
100-8-2.txt	ILS	627.7500	100	8	2	2581.7	5142.84	71.71	1953.95	5142.84	71.71	315.04	180.55	13.44	2568	316.17	1781708315
100-8-2.txt	VNS	627.7500	100	8	2	2592.77	10749.77	103.68	1965.02	10749.77	103.68	314.98	280.94	16.76	2568	358.26	1781717956
100-8-2.txt	SA	627.7500	100	8	2	2568	0	0	1940.25	0	0	44.19	0.62	0.79	2568	44.72	1781727787
100-8-4.txt	GRASP	654.3750	100	8	4	2187.77	14915.50	122.13	1533.39	14915.50	122.13	338.23	572.12	23.92	1947	344.24	1781729156
100-8-4.txt	ILS	654.3750	100	8	4	2114.6	18915.63	137.53	1460.23	18915.63	137.53	329.55	532.05	23.07	1871	355.66	1781739317
100-8-4.txt	VNS	654.3750	100	8	4	2129.7	12356.84	111.16	1475.33	12356.84	111.16	318.54	242.77	15.58	1920	308.76	1781749231
100-8-4.txt	SA	654.3750	100	8	4	1832.67	1760.57	41.96	1178.29	1760.57	41.96	44.84	0.63	0.80	1763	45.35	1781779243

APÊNDICE B.4: Resultados completos dos testes das instâncias geradas para o trabalho com 150 tarefas

Fonte: Autor (2026)

INSTANCIA	HEURISTICA	LB	J	M	R	MED_MAK	VARIANCIA_MAK	DVP_MAK	MEDIA_GAP	VAR_GAP	DVP_GAP	MEDIA_TEMPO	VAR_TEMPO	DVP_TEMPO	MELHOR_MAK	MELHOR_TEMPO	SEED_MELHOR_MAK
150-2-2.txt	GRASP	3401.5000	150	2	2	4212.4	193005.63	439.32	810.9	193005.63	439.32	670.48	3399.79	58.31	3451	824.19	1781888718
150-2-2.txt	ILS	3401.5000	150	2	2	3927.43	358385.36	598.65	525.93	358385.36	598.65	694.59	4187.95	64.71	3451	806.35	1781909159
150-2-2.txt	VNS	3401.5000	150	2	2	3997.3	268263.73	517.94	595.8	268263.73	517.94	650.85	2184.89	46.74	3451	668.11	1781956375
150-2-2.txt	SA	3401.5000	150	2	2	3451	0	0	49.5	0	0	86.08	121.20	11.01	3451	86.59	1781990719
150-2-4.txt	GRASP	4018.0000	150	2	4	4018.87	9.9126	3.15	0.87	9.91	3.15	760.29	2179.08	46.68	4018	768.81	1781780671
150-2-4.txt	ILS	4018.0000	150	2	4	4018.77	13.3575	3.65	0.77	13.36	3.65	762.19	6992.66	83.62	4018	743.66	1781803500
150-2-4.txt	VNS	4018.0000	150	2	4	4019.1	24.3690	4.94	1.1	24.37	4.94	714.95	5258.50	72.52	4018	666.57	1781826569
150-2-4.txt	SA	4018.0000	150	2	4	4018	0	0	0	0	0	85.63	77.03	8.78	4018	81.04	1781865649
150-4-2.txt	GRASP	1821.7500	150	4	2	4545.9	254377.96	504.36	2724.15	254377.96	504.36	698.03	6123.59	78.25	3950	805.32	1782077349
150-4-2.txt	ILS	1821.7500	150	4	2	4312.03	194660.79	441.20	2490.28	194660.79	441.20	664.08	2982.50	54.61	3950	613.46	1782124834
150-4-2.txt	VNS	1821.7500	150	4	2	4348.4	354709.56	595.57	2526.65	354709.56	595.57	643.41	1631.33	40.39	3950	604.22	1782145337
150-4-2.txt	SA	1821.7500	150	4	2	3950	0	0	2128.25	0	0	97.99	4.47	2.11	3950	96.20	1782164877
150-4-4.txt	GRASP	1849.7500	150	4	4	3280.13	31166.60	176.54	1430.38	31166.60	176.54	787.91	12386.98	111.30	2975	852.90	1782167905
150-4-4.txt	ILS	1849.7500	150	4	4	3297.43	54848.74	234.20	1447.68	54848.74	234.20	729.85	8838.52	94.01	2794	642.43	1782211404
150-4-4.txt	VNS	1849.7500	150	4	4	3243.93	45255.58	212.73	1394.18	45255.58	212.73	697.29	8529.55	92.36	2813	723.26	1782238547
150-4-4.txt	SA	1849.7500	150	4	4	2619.3	4137.25	64.32	769.55	4137.25	64.32	99.67	10.50	3.24	2544	96.79	1782265889
150-8-2.txt	GRASP	999.1250	150	8	2	5018.93	329170.62	573.73	4019.81	329170.62	573.73	674.19	3955.90	62.90	4411	622.39	1782268898
150-8-2.txt	ILS	999.1250	150	8	2	4629.8	221608.10	470.75	3630.68	221608.10	470.75	694.00	4789.06	69.20	4411	623.06	1782297778
150-8-2.txt	VNS	999.1250	150	8	2	4536.93	92521.51	304.17	3537.81	92521.51	304.17	648.20	3876.82	62.26	4411	707.09	1782336806
150-8-2.txt	SA	999.1250	150	8	2	4411	0	0	3411.88	0	0	101.79	4.83	2.20	4411	100.22	1782356279
150-8-4.txt	GRASP	911.1250	150	8	4	3260.03	20814.52	144.27	2348.91	20814.52	144.27	736.56	15467.59	124.37	2899	862.51	1782383188
150-8-4.txt	ILS	911.1250	150	8	4	3168.27	34857.31	186.70	2257.14	34857.31	186.70	746.99	12396.15	111.34	2665	800.96	1782405307
150-8-4.txt	VNS	911.1250	150	8	4	3214.63	43623.69	208.86	2303.51	43623.69	208.86	735.94	7445.02	86.28	2805	789.47	1782432469
150-8-4.txt	SA	911.1250	150	8	4	2420.4	3365.56	58.01	1509.28	3365.56	58.01	110.00	13.24	3.64	2323	112.75	1782470677

APÊNDICE C: COMPROVANTE DE APROVAÇÃO DO TRABALHO NO ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO (ENEPEP)

Capa do Artigo:	TN_WG_438_2166_54096_CAPA.pdf		
Artigo:	TN_WG_438_2166_54096.pdf		
Título:	GRASP, ILS E VNS APLICADOS AO PROBLEMA DE MÁQUINAS PARALELAS COM RESTRIÇÃO DE RECURSOS		
Tipo:	TN_WG - Workshop de Graduação		
Resultado:	Aprovado		
NOTAS (entre 1,0 e 5,0)			
Título (peso 1):	4,50	Revisão Bibl. (peso 1):	3,50
Resumo (peso 2):	4,50	Conclusões (peso 1):	4,00
Introdução (peso 1):	3,50	Relevância (peso 2):	3,00
Metodologia (peso 2):	4,00	Redação (peso 2):	3,50
Palavras Chave:	Máquinas paralelas idênticas, Restrição de recursos, Meta-heurísticas, Makespan, Sequenciamento		
Área:	03. PESQUISA OPERACIONAL		
Sub-área:	03.1. Modelagem, Simulação e Otimização		
Autores:	Vinicius Muniz Magalhães (535-MG), Aloisio de Castro Gomes Júnior (535-MG), caio shimomura (536-MG), Helton Cristiano Oliveira e Gomes (535-MG)		
Submetido:	18/05/2026 às 19:48:01 por Vinicius Muniz Magalhães		