UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA DE PRODUÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

BRUNO LINHARES DE SOUZA

APLICAÇÃO DE MÉTODOS E TÉCNICAS DE OTIMIZAÇÃO, SIMULAÇÃO E *MACHINE LEARNING* PARA RESOLUÇÃO DE UM PROBLEMA DE *FLOW SHOP* PERMUTACIONAL ESTOCÁSTICO

Recife
2025

BRUNO LINHARES DE SOUZA

APLICAÇÃO DE MÉTODOS E TÉCNICAS DE OTIMIZAÇÃO, SIMULAÇÃO E *MACHINE LEARNING* PARA RESOLUÇÃO DE UM PROBLEMA DE *FLOW SHOP* PERMUTACIONAL ESTOCÁSTICO

Dissertação de Mestrado apresentada ao Departamento de Engenharia de Produção da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção. Área de Concentração: Pesquisa Operacional.

Orientador: Prof. Dr. Raphael Harry Ribeiro Kramer

RECIFE

2025

.Catalogação de Publicação na Fonte. UFPE - Biblioteca Central

Souza, Bruno Linhares de.
Aplicação de métodos e técnicas de otimização, simulação e Machine Learning para resolução de um problema de flow shop permutacional estocástico / Bruno Linhares de Souza. - Recife, 2025.
141f.: il.

Dissertação (Mestrado) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Programa de Pós-Graduação em Engenharia de Produção, 2025.
Orientação: Prof. Dr. Raphael Harry Ribeiro Kramer.
Inclui referências e apêndices.

1. Flow shop Permutacional; 2. Otimização; 3. Meta-heurística; 4. Machine Learning; 5. Estocasticidade. I. Kramer, Raphael Harry Ribeiro. II. Título.

UFPE-Biblioteca Central

BRUNO LINHARES DE SOUZA

APLICAÇÃO DE MÉTODOS E TÉCNICAS DE OTIMIZAÇÃO, SIMULAÇÃO E *MACHINE LEARNING* PARA RESOLUÇÃO DE UM PROBLEMA DE *FLOW SHOP* PERMUTACIONAL ESTOCÁSTICO

Dissertação de Mestrado apresentada ao Departamento de Engenharia de Produção da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Mestre em Engenharia de Produção. Área de Concentração: Pesquisa Operacional.

Aprovada em: 25/02/2025.

BANCA EXAMINADORA

Prof. Dr. Raphael Harry Ribeiro Kramer
(Orientador)
Universidade Federal de Pernambuco

Profa. Dra. Isis Didier Lins (Examinadora Interna)
Universidade Federal de Pernambuco

Prof. Dr. Luciano Carlos Azevedo da Costa
(Examinador Externo)
Universidade Federal da Paraíba

AGRADECIMENTOS

Agradeço ao meu orientador Professor Raphael Kramer e a Fundação de Amparo à Ciência e Tecnologia de Pernambuco (FACEPE).

RESUMO

A crescente complexidade das transformações tecnológicas e mercadológicas, aliada ao aumento contínuo dos custos de produção, impõe a necessidade de buscar estratégias eficientes para o planejamento da produção. Neste cenário, otimizar a programação da produção torna-se essencial, especialmente em ambientes *flow shop*, onde a alocação eficiente de tarefas e recursos é um desafio constante. Este trabalho aborda um problema específico do tipo *Flow Shop* Permutacional, com foco em condições estocásticas, que envolvem incertezas nos tempos de processamento das operações. O *flow shop* permutacional, encontra-se na classe de problemas NP-difícil, e demanda constantemente a adoção de métodos de otimização, por ser estocástico requer a realização de estimativas (*e.g.* simulação e *machine learning*). Diante disso, este trabalho apresenta oito abordagens diferentes baseadas em otimização, aprendizagem por reforço, aprendizagem supervisionada e técnicas híbridas que combinam *Machine Learning* com as meta-heurísticas multi-start e GRASP. Esses métodos são relevantes porque, apesar de não garantirem a solução ótima, ajudam a encontrar boas soluções para problemas complexos, especialmente quando há incertezas envolvidas. Os testes realizados, com 110 instâncias diferentes, mostraram que os métodos baseados na meta-heurística GRASP não apresentaram bons resultados. No entanto, foi possível identificar que dentre as abordagens propostas a escolha do método mais adequado depende das características específicas de cada problema, como as dimensões das instâncias e os tempos de processamento. Os métodos desenvolvidos deram origem a um simulador WEB que pode ser utilizado para ajudar na programação da produção em unidades fabris. Essa ferramenta foi criada para auxiliar na tomada de decisões, oferecendo uma maneira prática de lidar com a complexidade dos problemas de *flow shop* permutacional sob condições estocásticas.

Palavras-chave: *Flow shop* Permutacional. Otimização. Meta-heurística. *Machine Learning*. Estocasticidade.

ABSTRACT

The increasing complexity of technological and market transformations, combined with the continuous rise in production costs, necessitates the need to seek efficient strategies for production planning. In this context, optimizing production scheduling becomes essential, especially in flow shop problems, where the efficient allocation of tasks and resources is a constant challenge. This dissertation addresses a specific type of Permutation Flow Shop problem, focusing on stochastic conditions involving uncertainties in operation processing times. The permutation flow shop problem is classified as an NP-hard problem and continuously requires the adoption of optimization methods. Due to its stochastic nature, it demands estimations (e.g., simulation and machine learning). In this regard, this research presents eight different approaches based on optimization, reinforcement learning, supervised learning, and hybrid techniques that combine machine learning with multi-start metaheuristics and GRASP. These methods are relevant because, although they do not guarantee the optimal solution, they help to find good solutions for complex problems, especially when uncertainties are involved. Tests conducted on 110 different instances showed that methods based on the GRASP metaheuristic did not yield good results. However, it was possible to identify that, among the proposed approaches, the choice of the most suitable method depends on the specific characteristics of each problem, such as instance sizes and processing times.The developed methods led to the creation of a web simulator that can be used to assist in production scheduling in manufacturing units. This tool was designed to aid decision-making, offering a practical way to handle the complexity of permutation flow shop problems under stochastic conditions.

Keywords: Permutational Flow Shop; Optimization; Metaheuristics; Machine Learning; Stochasticity.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

ABC	*Artificial Bee Colony*
AHP	*Analytic Hierarchy Process*
EMA	Erro Médio Absoluto
EPE	Empresa de Pesquisa Energética
FSP	*Flow shop Scheduling Problem*
GA	*Genetic Algorithm*
GRASP	*Greedy Randomized Adaptive Search Procedure*
GRASP-RLM-POOL	GRASP com Regressão Linear Múltipla*(pool)*
GRASP-SIMH	GRASP - Simheurística
GRASPMED	GRASP com Tempos Médios
IEA	*International Energy Agency*
ILS	*Iterated Local Search*
INPI	Instituto Nacional de Propriedade Industrial
IQL	*Improvement Q-Learning*
IQL-S	*Improvement Q-Learning - Simulation*
LCR	Lista de Candidatos Restrita
ML	*Machine Learning*
MM	Melhor *Makespan*
MMC	*Makespan* Monte Carlo
MSIQLS-S	*Multi-start Improvement Q-Learning - Simulation*
MSNEHT-RLM-POOL	*Multi-start* com NEHT e Regressão Linear Múltipla*(pool)*
MSNEHT-SIMH	*Multi-star*t NEHT - Simheurística
MSQLNEH-S	*Multi-start Q-Learning NEH - Simulation*
MTPM	Matriz de Tempos de Processamento Médio
NEH	Nawaz-Enscore-Ham
NEHT	Nawaz-Enscore-Ham-Taillard
NEHTBLMED	NEHT com Busca Local e Tempos Médios
PAES	*Pareto Archived Evolution Strategy*
PCA	*Principal Component Analysis*
PFSP	*Permutation Flow shop Scheduling Problem*
PFSPST	*Permutation Flow shop Scheduling Problem with Stochastic Times*

PPO	*Proximal Policy Optimization*
PSO	*Particle Swarm Optimization*
QL	*Q-Learning*
QLNEH	*Q-Learning NEH*
QLNEH-S	*Q-Learning NEH - Simulation*
REMQ	Raiz do Erro Médio Quadrático
RLM	Regressão Linear Múltipla
SA	*Simulated Annealing*
SMC	Simulação de Monte Carlo
SVM	*Support Vector Machine*
TS	*Tabu Search*
VNS	*Variable Neighborhood Search*

SUMÁRIO

1 INTRODUÇÃO

Em um ambiente de constantes transformações tecnológicas e mercadológicas, cresce a necessidade das organizações por um planejamento bem definido acerca das operações e capacidade de produção. Nessa perspectiva, estabelecer adequadamente o planejamento da produção apresenta-se como um mecanismo fundamental na capabilidade produtiva, fornecendo o suporte necessário para o atendimento das variações da demanda e alocação de recursos (Guzman *et al.*, 2022).

No curto prazo, pode-se orientar a programação da produção ao sequenciamento (*scheduling*) de tarefas, visando otimizar o tempo de processamento global das tarefas. No contexto da indústria 4.0, com todos os recursos tecnológicos digitais disponíveis, têm-se viabilidade do desenvolvimento de tecnologias capazes de promover auxílio na definição e estruturação de uma política de produção (Oluyisola *et al.*, 2022).

Os problemas de *scheduling* apresentam diversas variantes (*e.g.*, envolvendo uma única máquina ou múltiplas máquinas), dentre as quais destaca-se a variante *flow shop* (FSP, do inglês *Flow shop Scheduling Problem*), comumente encontrada em empresas de manufatura, têxtil, automotiva, etc. (Fuchigami, 2015). O problema consiste em determinar o sequenciamento de n de tarefas em m máquinas, onde todas as tarefas devem ser processadas nas m máquinas seguindo o mesmo roteiro ($maq_1, maq_2, \cdots, maq_m$) (Nagarajan e Sviridenko, 2008).

A resolução do FSP envolvendo três ou mais máquinas demanda um alto custo computacional e enquadra-se na classe de problemas NP-difícil (Dudek *et al.*, 1992). Frente a isso, os métodos existentes capazes de encontrar a solução ótima, denominados de algoritmos exatos, nem sempre são adequados para resolução, pois demandam um tempo computacional tão alto que muitas vezes torna inviável a sua utilização, principalmente para instâncias maiores. Deste modo, abordagens envolvendo algoritmos evolucionários, heurísticos e meta-heurísticos são mais utilizados para resolver problemas de sequenciamento (Liang *et al.*,2022). Embora esses algoritmos não garantam a solução ótima, são capazes de obter uma boa solução em um tempo computacional curto.

O Problema de Sequenciamento *Flow shop* Permutacional (PFSP, do inglês *Permutation Flow shop Scheduling Problem*), refere-se a uma variante do *flow shop* na qual n tarefas (*jobs*) passam obrigatoriamente em uma quantidade m de máquinas em uma mesma sequência, ou seja, as tarefas processadas em uma máquina seguem para próxima máquina rigorosamente na mesma ordem que fora processada na máquina anterior (Ravetti *et al.*,2012).

A literatura propõe uma série de métricas que podem ser otimizadas em um problema *flow shop*, tais como: tempo total de *setup*, tempo total de atraso, tempo de término da última tarefa

na última máquina (*makespan*), entre outras. Inclusive, alguns trabalhos possuem abordagens em caráter multiobjetivo, ou seja, visam otimizar mais de uma métrica, sendo muitas vezes os objetivos conflitantes entre si (Li e Li, 2015).

Em aplicações reais, o tempo de processamento de uma tarefa em uma máquina dificilmente será constante, haja vista que diversos fatores podem impactar no tempo de execução da tarefa. Em outras palavras, o tempo de processamento da tarefa i na máquina j será um valor aleatório. Nesta perspectiva, nota-se que os referidos tempos de execução, possuem naturalmente um caráter estocástico, portanto, dotados de uma incerteza associada. Dessa forma, a tarefa pode ser finalizada em um tempo anterior ou posterior ao esperado. Uma forma de lidar com a estocasticidade dos tempos é utilizando simulações computacionais. Assim sendo, na posse de dados referentes a variabilidade dos tempos de processamentos das tarefas, pode-se desenvolver um algoritmo capaz de simular tempos de processamento estocásticos, no contexto do FSP, obtendo-se as variações possíveis para execução de uma determinada tarefa em uma respectiva máquina.

A aprendizagem de máquina (ML, do inglês *Machine Learning*) tem como principal característica buscar padrões nos dados, através de funções matemáticas, objetivando a extração de conhecimento e fazer previsões, tomando por base os dados de entrada. Há várias técnicas e abordagens nos escopos da ML, cada uma adequada a diferentes tipos de problemas e conjuntos de dados. O que a habilita para aplicação em estudos desenvolvidos no contexto do PFSP, haja vista a possibilidade de acesso a dados relacionados aos processos produtivos fabris.

As técnicas de ML têm sido aplicadas em diversas áreas, tais como: saúde, *streaming*, varejo, políticas públicas, mercado financeiro e na indústria (Zhong *et al.*, 2016; Barman *et al.*, 2019; Amarasinghe *et al.*, 2023). Em cada uma dessas áreas, os algoritmos oferecem oportunidades para melhorar a eficiência e possibilitar uma tomada de decisão mais precisa e personalizada. O uso das ferramentas de ML apresenta um grande potencial de crescimento à medida que as organizações reconhecem o valor agregado que a incorporação dessa tecnologia pode proporcionar aos seus processos operacionais e de tomada de decisão, principalmente no contexto atual de um mundo cada vez mais digital e orientado por dados (Shafiq, 2024).

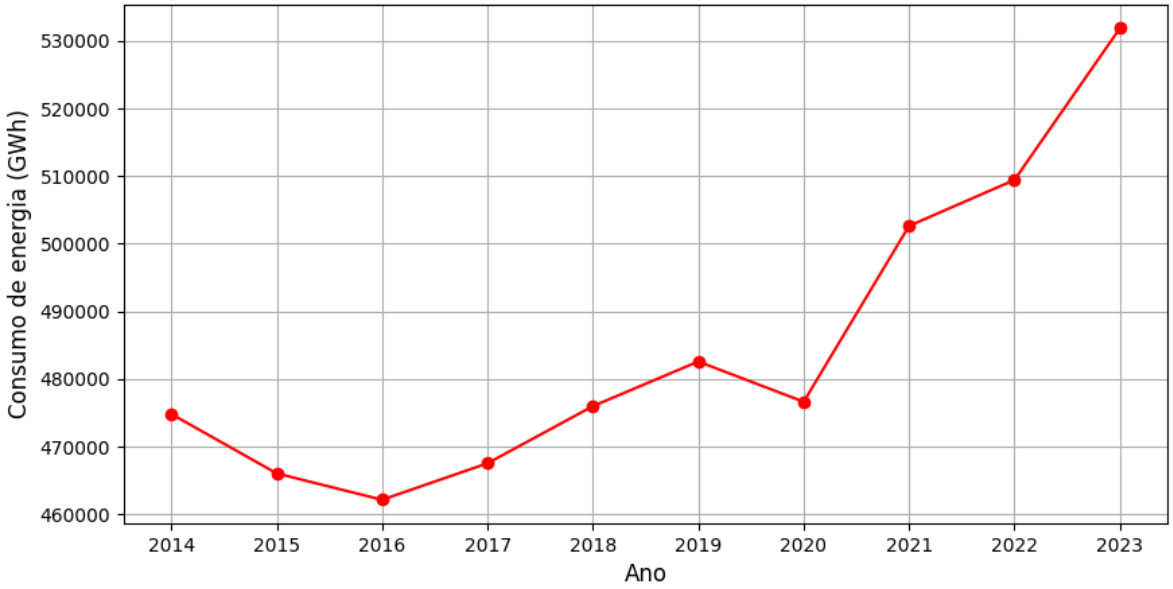
Diante do exposto, a adoção de uma metodologia capaz de integrar as técnicas de *machine learning*, otimização e simulação, torna possível a criação de sistemas mais inteligentes e adaptativos que possam gerenciar o fluxo de produção em um ambiente de *flow shop*. Essas abordagens não apenas melhoram a eficiência operacional, mas também permitem uma resposta mais rápida a mudanças nas condições do mercado, demanda do cliente e disponibilidade de recursos. Em última análise, isso leva a uma produção mais ágil, econômica e sustentável. Neste propósito, a pesquisa objetiva responder a

seguinte pergunta: **como unir os algoritmos heurísticos e as técnicas de *machine learning* para reduzir o tempo de resolução do PFSP estocástico e gerar soluções de boa qualidade?**

1.1 JUSTIFICATIVA E RELEVÂNCIA

Em 2023, só no Brasil foram consumidos aproximadamente 532.000 GWh de energia elétrica, resultando em um aumento de 4,4 % em relação a 2022. Desse total, 188.476 GWh foram consumidos pela indústria, pontuando um aumento de 2,2 % também em relação ao ano anterior (EPE, 2024). A *International Energy Agency* (IEA) projeta um crescimento de pelo menos 3,4% no período de 2024-2026 na demanda mundial por energia (IEA, 2024). A Figura 1 demonstra o consumo de energia no Brasil, entre o período de 2014 a 2023, baseando-se nos dados da Empresa de Pesquisa Energética (EPE).

Figura 1 – Consumo de energia no Brasil (2014-2023)



Fonte: Adaptado de EPE (2024)

Outro ponto a ser considerado, refere-se aos custos de produção que também sofreram um forte aumento por influências geopolíticas e da pandemia do COVID-19 (Allam *et al.*, 2022). Dado o aumento na demanda por energia e dos custos crescentes de produção, as empresas enfrentam pressões adicionais para melhorar a eficiência de seus processos industriais. Consequentemente, procurar estratégias que visam otimizar a produção e reduzir custos é algo de suma importância para as organizações.

Logo, a relevância do estudo do FSP e do PFSP está intrinsecamente ligada à busca por eficiência e competitividade nas operações de produção. O desenvolvimento de estratégias de sequenciamento robustas e eficazes para lidar com a incerteza nos tempos de processamento, podem

melhorar significativamente a utilização de recursos, reduzir os tempos de ciclo, aumentar a qualidade e a confiabilidade dos processos produtivos.

Neste contexto, resolver o PFSP de maneira eficiente emerge como um desafio significativo a ser enfrentado pelas empresas, pois uma programação ineficiente pode resultar em tempos de produção mais longos, aumento nos custos operacionais e menor competitividade no mercado.

Um aspecto central na resolução do PFSP engloba o alto custo computacional demandado, haja vista que esse encontra-se na classe NP-difícil. Nesta perspectiva, o uso de algoritmos heurísticos e meta-heurísticos são imprescindíveis para auxiliar na resolução do PFSP, pois mesmo não garantindo a otimalidade, possibilitam a obtenção de boas soluções em um tempo computacional razoável.

Notoriamente, em aplicações reais, os tempos de processamento de uma tarefa em uma ou várias máquinas estão sujeitos a variações e intercorrências que podem impactar o tempo de execução de uma tarefa. Deste modo, o tempo real de processamento é dotado de incerteza, assim assumindo uma forma estocástica.

A variabilidade nos tempos de processamento pode acarretar em atrasos na produção, o que potencialmente pode impactar negativamente toda cadeia produtiva, haja vista que tais atrasos são capazes de gerar um efeito cascata, comprometendo o cumprimento dos prazos de entrega de produtos e, consequentemente, afetando a satisfação do cliente. Além disso, a ineficiência resultante pode aumentar os custos operacionais, devido à necessidade de horas extras ou reprogramações, e reduzir a utilização eficiente dos recursos disponíveis, levando a uma queda na produtividade geral.

Ainda nessa conjuntura, o emprego de técnicas de ML podem ser exploradas com intuito de tornar mais rápido a realização de uma estimativa do *makespan*, haja vista que utilizar métodos de simulações com muitas iterações pode tornar a estimativa do *makespan* computacionalmente custosa. Deste modo, aplicar métodos de aprendizagem de máquina pode auxiliar a encontrar valores aproximados de makespan após a construção de uma sequência e com o uso de métodos meta-heurísticos, pode-se melhorar a solução inicial, conduzindo a busca na perspectiva de encontrar um sequenciamento mais próximo da realidade e capaz de minimizar o *makespan*.

Isto posto, compreende-se a importância de abordar soluções para FSP e PFSP considerando os aspectos da estocasticidade. Pois, por um lado aumenta a complexidade do problema, mas em contrapartida fornece uma maior proximidade com a realidade, consequentemente indicando resultados mais realistas. Nesse ponto, conflui o uso de simulação computacional, visto que com o emprego desta técnica é possível lidar com a incerteza nos tempos de processamento, ou seja, pode-se simular variações nos tempos de processamento de uma tarefa em uma máquina e consequentemente, analisar os impactos nos mais diversos indicadores de desempenho.

Perante ao exposto, este trabalho visa recorrer as técnicas de ML para auxílio em estimar o *makespan* esperado. Visando encontrar soluções de melhor qualidade frente ao uso de estratégias de otimização isoladas. Propositivamente usando modelos baseados na Regressão Linear Múltipla, haja vista a capacidade de gerar uma previsão inicial rapidamente, assim como explorar abordagens baseadas em aprendizagem por reforço, mais especificamente no que tange a métodos explorado na literatura, embasados no algoritmo *Q-Learning*.

1.2 OBJETIVOS

A seguir serão apresentados os objetivos do trabalho, os quais estão divididos explicitamente em Objetivo Geral e Objetivos Específicos.

1.2.1 Objetivo Geral

O objetivo principal do trabalho consiste em propor e implementar algoritmos de otimização combinados com técnicas de simulação e *machine learning* para resolução de um PFSP com tempos de processamento estocásticos.

1.2.2 Objetivos Específicos

Para atingir o objetivo principal, foram definidos os seguintes objetivos específicos:

- Revisar a literatura referente aos problemas de sequenciamento da produção que abordem questões de natureza estocástica;
- Propor um novo método para calcular a simulação estocástica, no contexto do PFSP;
- Construir métodos híbridos que utilizem técnicas de simulação, meta-heurística e ML para resolução do problema de pesquisa;
- Implementar algoritmos de otimização para resolução do problema de pesquisa;
- Realizar uma análise comparativa entre as diferentes versões dos algoritmos implementados;
- Desenvolver um simulador WEB, capaz de prever o *makespan* esperado em ambientes PFSP com considerações estocásticas.

1.3 ORGANIZAÇÃO DO TRABALHO

O restante do trabalho está organizado da seguinte forma. O Capítulo 2 apresenta o referencial teórico. No Capítulo 3, mostra a revisão da literatura. O capítulo 4 explana a descrição

do problema da pesquisa. O capítulo 5 expõe a metodologia do trabalho. O Capítulo 6 detalha os resultados e discussões . Por fim, o Capítulo 7 apresenta as conclusões e sugestões de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste capítulo serão abordados quatro tópicos relativos à área temática do projeto de pesquisa, dividindo-se em: *flow shop* permutacional e questões estocásticas; algoritmos de otimização; algoritmos de *machine learning*; e iv) testes estatísticos.

2.1 PROBLEMA DE SEQUENCIAMENTO

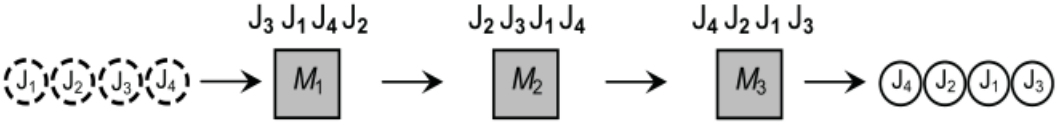
O sequenciamento é um processo decisório recorrente em diversas indústrias, que envolve a alocação de recursos para a execução de tarefas ao longo de períodos específicos. Seu objetivo principal é otimizar (minimizar ou maximizar) um ou mais critérios de desempenho (Pinedo, 2016). Em ambientes industriais, as atividades a serem realizadas podem ser divididas em um conjunto de tarefas ou operações, onde cada uma representa uma parte do trabalho (Emmons e Vairaktarakis, 2013).

Segundo Fuchigami (2015), a classificação de problemas envolvendo programação da produção, envolve:

- **Máquina única:** caracteriza-se pela existência de somente uma etapa de produção;
- **Máquinas paralelas:** possui apenas um estágio de produção, no entanto contendo mais de duas máquinas disponíveis para executar qualquer tarefa (máquinas paralelas);
- ***Flow shop*:** as máquinas não são iguais e estão dispostas em série. Seguindo uma rota unidirecional que as tarefas seguem. Todas possuem o mesmo fluxo de processamento nas máquinas;
- ***Flow shop* híbrido ou flexível:** as tarefas são processadas em vários estágios de produção, sendo na mesma ordem e em cada um deles há máquinas paralelas. Cada tarefa é processada por uma única máquina em cada estágio;
- ***Job shop*:** cada tarefa segue um fluxo individual;
- ***Job shop* flexível:** a distinção entre o *Job Shop* tradicional é que existem estágios com mais de uma máquina paralela;
- ***Open shop*:** é similar ao *job shop*, no entanto as tarefas não possuem rotas específicas.

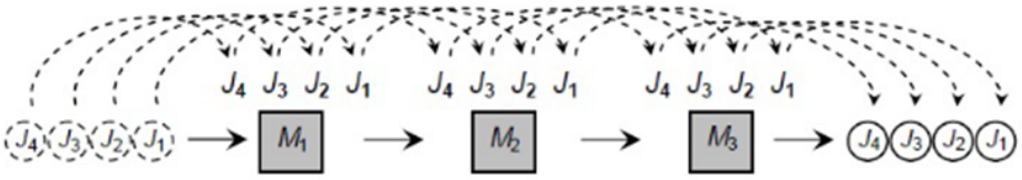
Uma das variantes do problema de sequenciamento *flow shop* é o *flow shop* permutacional, diferindo do *flow shop* no campo em que as tarefas precisam ser processadas na mesma ordem que foram processadas na máquina anterior (Chakraborty, 2009). A Figura 2 ilustra a representação do *flow shop* não permutacional, enquanto a Figura 3, representa o *flow shop* permutacional. Em ambas figuras, têm-se um conjunto de quatro tarefas processadas em um conjunto de três máquinas.

Figura 2 – Representação do *flow shop*



Fonte: Fuchigami (2015)

Figura 3 – Representação do *flow shop* permutacional



Fonte: Hordones e Fuchigami (2017)

2.1.1 Problema do Sequenciamento Flow shop Permutacional Estocástico

O arranjo do FSP e PFSP acarreta em um desafio significativo de programação e sequenciamento de tarefas para maximizar a eficiência do processo produtivo. Assim, resolver o PFSP implica em encontrar um sequenciamento ideal das tarefas que minimize uma métrica de desempenho específica, por exemplo, o *makespan* ($C_{\text{máx}}$). No entanto, devido à natureza estocástica dos tempos de processamento das tarefas, o PFSP se torna um desafio ainda mais complexo (Maccarrone *et al.*, 2018). Neste cenário, existe uma generalização do PFSP, denominada Problema Sequenciamento *Flow shop* Permutacional com Tempos Estocásticos (PFSPST, do inglês *Permutation Flow shop Scheduling Problem with Stochastic Times*), onde o tempo de processamento de uma tarefa i em uma máquina j não é tratado como um valor determinístico e sim como uma variável aleatória que segue uma distribuição de probabilidade e assim introduzindo incerteza sobre o tempo de finalização de uma tarefa (Juan *et al.*, 2014). Para isso, uma metodologia comumente adotada, envolve a aplicação de técnicas de simulação computacional, baseadas em distribuições probabilísticas com o intuito de gerar tempos de processamento estocásticos (Juan *et al.*, 2014; Gonzalez-Neira *et al.*, 2017; Gonzalez-Neira *et al.*, 2021), isto permite avaliar o desempenho em diferentes sequenciamentos de tarefas sob condições de incerteza, possibilitando a identificação de políticas de sequenciamento que minimizem os impactos da variabilidade nos tempos de processamento reais.

2.2 HEURÍSTICAS E META-HEURÍTICAS

A resolução (exata) de problemas de otimização combinatória que estejam em NP-difícil demandam um alto custo computacional (Qiu *et al.*, 2022). Consequentemente, o uso de estratégias

heurísticas é recorrente. Há uma diversidade de algoritmos que são classificados como heurísticos, estes ainda podem ser subdivididos em dois grandes grupos: heurísticas construtivas e heurísticas de refinamento. Os primeiros relacionados à construção de uma solução inicial viável. Já as heurísticas de refinamento, visam melhorar soluções existentes, baseando-se em técnicas de exploração de vizinhança de uma solução. A construção de uma solução inicial viável, por exemplo, pode ser estabelecida de forma completamente aleatória ou por meio de algoritmos gulosos, que constroem uma solução elemento a elemento, levando em consideração o melhor ganho momentâneo, ou seja uma solução localmente ótima (Cormen *et al.*, 2022). Como exemplo de algoritmo guloso podemos citar a heurística já consolidada Nawaz-Enscore-Ham (NEH) (Nawaz *et al.*, 1983).

Ainda, há um conjunto de algoritmos de otimização, cuja a estrutura mais robusta compondo um conjunto de estratégias para o desenvolvimento da otimização heurística, chamado de meta-heurística (Sörensen e Glover, 2013). Segundo Osman e Kelly (1996) uma meta-heurística pode ser definida formalmente como um processo iterativo, capaz de orientar uma heurística e que combina conceitos de maneira inteligente, visando explorar os espaços de busca e utilizando estratégias de aprendizagem para estruturar informações para encontrar soluções eficientes e próximas do ótimo.

São exemplos de algoritmos meta-heurísticos clássicos: *Simulated Annealing* (SA) (Kirkpatrick *et al.*, 1983), Busca Tabu (TS, do inglês *Tabu Search*) (Glover, 1989), *Greedy Randomized Adaptive Search Procedure* (GRASP) (Feo e Resende, 1995), Algoritmos Genéticos (GA, do inglês *Genetic Algorithms*) (Goldberg,1989), Busca em Vizinhança Variável (VNS,do inglês *Variable Neighborhood Search*) (Mladenović e Hansen,1997), entre outros.

2.2.1 Algoritmos Construtivos Nawaz-Enscore-Ham e Nawaz-Enscore-Ham-Taillard

Dentre os métodos construtivos mais conhecido no contexto de problemas FSP e PFSP, encontra-se a heurística Nawaz-Enscore-Ham (NEH), desenvolvida por Nawaz *et al.* (1983), cuja a finalidade traduz-se em sequenciar uma lista de *n* tarefas em *m* máquinas com intuito de minimizar o *makespan*, e que resumidamente pode ser descrita nos passos a seguir:

1º Passo: calcular a soma dos tempos de processamento de cada tarefa em todas as máquinas. Seguindo-se de um ordenamento não-crescente das tarefas de acordo com a soma dos tempos de processamento;

2º Passo: as duas primeiras tarefas com maior soma dos tempos de processamento formam a subsequência inicial, cuja a ordem dessas tarefas é definida com base no arranjo que resultar no menor *makespan*;

3º Passo: as demais tarefas são inseridas uma a uma na subsequência, testando-a em todas

as posições possíveis entre as tarefas já armazenadas na subsequência, finalmente alocando-a na posição da subsequência que apresentar o menor valor para o *makespan* e consolidando o sequenciamento ao inserir as n tarefas na lista.

Uma extensão da heurística NEH foi desenvolvida na década de 90 e ficou conhecida como Nawaz-Enscore-Ham-Taillard (NEHT), a heurística publicada por Taillard (1990), descreve um método que utiliza operações matriciais para realização da heurística construtiva NEH, tornando-a mais eficiente. O método objetiva calcular o $C_{\text{máx}}$ após a inserção da tarefa k na $i-$ésima posição e pode ser explicada em quatro passo (Taillard, 1990):

1º Passo: calcular o tempo de início da $i-$ésima tarefa na $j-$ésima máquina, no artigo original denominado e_{ij}.

$$\begin{cases} e_{0j} = 0, \quad e_{i0} = 0, \\ e_{ij} = \text{máx}(e_{i,j-1}, e_{i-1,j}) + t_{ij}, \\ i = 1, \ldots, k-1 \quad \text{e} \quad j = 1, \ldots, m, \quad \forall \quad i, j \end{cases} \tag{2.1}$$

2º Passo: calcular o tempo de duração entre a $i-$ésimo tarefa na $j-$ésima máquina e o final da operação (q_{ij}).

$$\begin{cases} q_{kj} = 0, \quad q_{i,m+1} = 0, \\ q_{ij} = \text{máx}(q_{i,j+1}, q_{i+1,j}) + t_{ij}, \\ i = k-1, \ldots, 1 \quad \text{e} \quad j = m, \ldots, 1, \quad \forall \quad i, j \end{cases} \tag{2.2}$$

3º Passo: calcular o tempo de conclusão relativo mais antigo (f_{ij}) na $j-$ésima máquina da tarefa k inserida na $i-$ésima posição.

$$\begin{cases} f_{i0} = 0 \\ f_{ij} = \text{máx}(f_{i,j-1}, e_{i-1,j}) + t_{kj} \\ i = 1, \ldots, k \quad \text{e} \quad j = 1, \ldots, m, \quad \forall \quad i, j \end{cases} \tag{2.3}$$

4º Passo: calcular o valor do $C_{\text{máx}}$ parcial (M_i) ao adicionar a tarefa k na $i-$ésima posição

$$\begin{cases} M_i = \text{máx}(f_{i,j} + q_{ij}) \\ i = 1, \ldots, k \quad \text{e} \quad j = 1, \ldots, m, \quad \forall \quad i, j \end{cases} \tag{2.4}$$

Ainda segundo o autor, o algoritmo pode executar todos as etapas descritas em $\mathcal{O}(km)$, enquanto a heurística NEH original executa em complexidade $\mathcal{O}(n^2m)$.

2.2.2 Busca Local

A busca local em algoritmos heurísticos é uma técnica que busca encontrar soluções melhores em um espaço de busca limitado, mediante a exploração da vizinhança dada uma solução (Amaral *et al.*, 2021), ou seja, em vez de explorar todo o espaço de busca (o que pode ser computacionalmente caro), a busca local foca em fazer pequenas alterações na solução atual para encontrar soluções melhores. Uma busca local para ser considerada efetiva deve ser capaz de encontrar mínimos ou máximos locais (Blum e Roli,2003). O Algoritmo 1, mostra o pseudocódigo genérico de uma busca local com objetivo de minimização, no qual, *sol_inicial* representa uma solução inicial viável, s representa a melhor solução corrente, $I(s)$ é o conjunto de soluções na vizinhança de s e $f(.)$ a função que calcula o custo de uma solução.

A definição de uma vizinhança ocorre por meio de estruturas, denominadas operadores de vizinhança, que transforma uma solução em outra solução próxima (Blum e Roli,2003). Eles são fundamentais para a exploração do espaço de soluções, permitindo que se avaliem variações em uma solução atual. No contexto do PFSP, existem diversos operadores de vizinhança. Por exemplo, Talbi (2009):

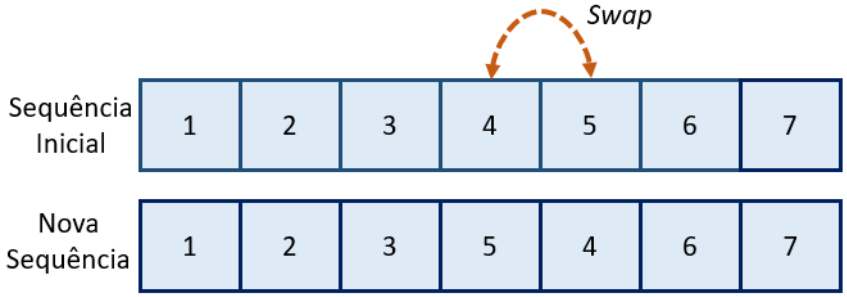
- ***Swap***: efetua uma troca entre duas tarefas da sequência;
- **Inserção**: um elemento da sequência é removido e colocado em outra posição.

Ainda, podemos citar o *2-opt* operador comumente aplicado na resolução do problemas do caixeiro viajante (Lin, 1965), também utilizado para modificação de vizinhança no PFSP (Zaied *et al.*, 2021). Vale ressaltar que, a escolha adequada dos operadores pode influenciar significativamente a capacidade do algoritmo de encontrar boas soluções. A Figura 4 ilustra a exemplificação do operador de vizinhança *swap*, enquanto a Figura 5 apresenta a ilustra do operador de vizinhança *2-opt*.

Algoritmo 1: Procedimento BuscaLocal(*sol_inicial*)

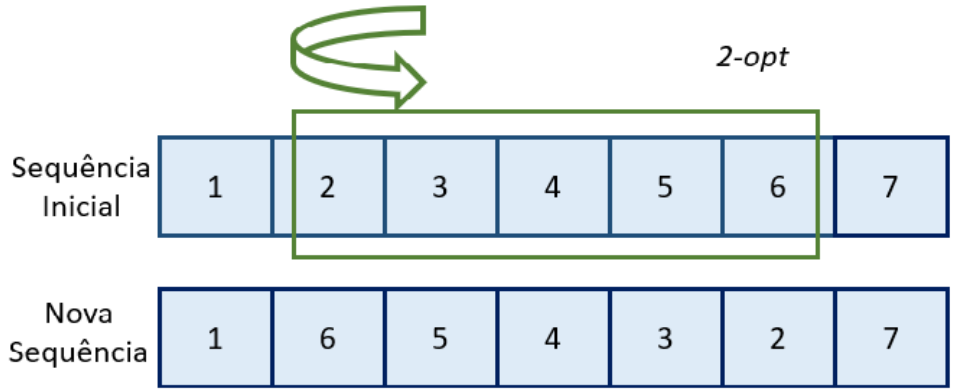
```
1:  s = sol_inicial
2:  f(s) = f(sol_inicial)
3:  while Enquanto o critério de parada não for atendido do
4:      faça s' ∈ I(s)
5:      calcule f(s')
6:      if f(s') < f(s) then
7:          s ← s'
8:          f(s) ← f(s')
9:      end if
10: end while
11: return S, f(S)
```

Figura 4 – Representação do operador de vizinhança *swap*



Fonte: Elaborada pelo autor (2025)

Figura 5 – Representação do operador de vizinhança *2-opt*



Fonte: Elaborada pelo autor (2025)

A busca local pode ser desenvolvida, por exemplo, na perspectiva de duas estratégias. A primeira é chamada de Melhor Melhoria (em tradução livre do inglês, *Best Improvement*), que visa avaliar todas as soluções vizinhas e escolher a melhor solução entre todas, e a segunda, denomina-se Primeira Melhoria (em tradução livre do inglês, *First Improvement*), que seleciona a primeira solução melhorada encontrada no processo de busca local (Breedam, 2001).

2.2.3 Meta-heurística Multi-start

A *Multi-start* é uma técnica de otimização projetada para orientar a construção de soluções no espectro do procedimento de busca local (Glover e Kochenberger, 2003). O princípio fundamental do algoritmo é realizar várias execuções de buscas locais a partir de diferentes soluções iniciais, buscando explorar diferentes regiões do espaço de soluções e, assim, aumentar a probabilidade de encontrar o ótimo global. Este método se destaca por sua simplicidade na exploração de soluções, o que o torna uma abordagem popular para problemas de otimização combinatória e contínua.

O Algoritmo 2 apresenta o pseudocódigo da meta-heurística *multi-start*, onde x^* é a melhor solução encontrada, N é o número de iterações ou *starts* de busca da meta-heurística, $f(.)$ é a função que calcula o custo de uma solução, $\mathscr{F}$ é o conjunto de soluções viáveis, x_i é uma solução inicial

aleatória viável, x_{local} é a solução corrente encontrada na busca local e x^* a melhor solução encontrada.

Algoritmo 2: Multi-start(N, $f(.)$, $\mathscr{F}$)

1: $x^* \leftarrow []$
2: $f(x^*) \leftarrow \infty$
3: **for** $i = 1$ **to** N **do**
4: Gerar solução inicial aleatória $x_i \in \mathscr{F}$
5: Executar BuscaLocal(x_i) para obter x_{local}
6: **if** $f(x_{\text{local}}) < f(x^*)$ **then**
7: $x^* \leftarrow x_{\text{local}}$
8: $f(x^*) \leftarrow f(x_{\text{local}})$
9: **end if**
10: **end for**
11: **return** x^*, $f(x^*)$

2.2.4 Meta-heurística GRASP

A meta-heurística GRASP representa uma estratégia de amostragem interativa, onde cada iteração gera uma solução para o problema em questão, sendo mantida como resultado final a solução que apresentar melhor resultado frente todas as iterações do GRASP (Feo e Resende, 1995).

O procedimento,basicamente envolve duas etapas. A primeira está relacionada a implementação de uma solução construtiva, enquanto a segunda está relacionada a realização de uma busca local (Glover e Kochenberger, 2003). Um dos componentes mais representativo da heurística construtiva é a Lista de Candidatos Restrita (LCR), composta pelos melhores elementos candidatos a comporem a solução inicial (em construção), onde um elemento é escolhido aleatoriamente a cada iteração. Ressalta-se que o benefício de escolher cada elemento (e, consequentemente, a LCR) é atualizado(a) a cada iteração da construção, de modo a refletir as mudanças relacionadas a escolha do elemento anterior (Feo e Resende, 1995), subordinada a um parâmetro probabilístico $\lambda \in [0,1]$. Este controla o grau de gulosidade da solução construtiva:

- Quando λ está **próximo de 0**, a construção da solução é mais gulosa, ou seja, a cada iteração a solução será formada escolhendo a melhor opção disponível com base no critério de avaliação (por exemplo, minimizando o custo).
- Quando λ está **próximo de 1**, a construção da solução se torna mais aleatória, permitindo que a solução explore mais opções, mesmo que não sejam as melhores.

Além do parâmetro λ a meta-heurística GRASP demanda o número de iterações N e a função $f(.)$ que calcula o custo de uma solução.

O Algoritmo 3 representa o pseudocódigo da estrutura da meta-heurística GRASP, enquanto o Algoritmo 4 apresenta o pseudocódigo genérico da solução construtiva, baseada na LCR. Ressaltando que x é a solução construtiva, $x*$ a melhor solução encontrada e x_{local} é a solução corrente encontrada na busca local.

Algoritmo 3: GRASP(N, $f(.)$, λ)

```
1:  x* ← []
2:  f(x*) ← ∞
3:  for i = 1 to N do
4:      x ← ConstrutivaGRASP(f(.), λ)
5:      x_local ← BuscaLocal(x)
6:      if f(x_local) < f(x*) then
7:          x* ← x_local
8:          f(x*) ← f(x_local)
9:      end if
10: end for
11: return x*, f(x*)
```

Algoritmo 4: ConstrutivaGRASP(λ)

```
1: x ← []
2: LCR ← []
3: while x não estiver completa do
4:     Construir LCR com base no parâmetro λ
5:     Selecionar aleatoriamente um elemento e ∈ LCR
6:     x ← x ∪ e
7:     Atualizar LCR
8: end while
9: return x
```

2.3 SIMULAÇÃO DE MONTE CARLO

A simulação corresponde a um processo de experimentação a partir de um modelo detalhado que represente um sistema real, visando tornar possível experimentar os comportamentos do sistema a modificações em sua estrutura ou condições de contorno (Bowden *et al.*, 2013).

Dentre as abordagens mais comuns de simulação encontra-se a Simulação de Monte Carlo (SMC), método criado por John von Neumann e Stanislaw Ulam durante projetos de pesquisa envolvendo armas nucleares no período da Segunda Guerra Mundial. No contexto, os pesquisadores precisavam entender o comportamento de sistemas complexos e incertos, como a dispersão aleatória de nêutrons em materiais nucleares (Rubinstein e Kroese, 2017), desenvolvendo um método estatístico

para esse fim (Metropolis e Ulam, 1949). A SMC é uma técnica estatística amplamente utilizada para modelar e analisar sistemas complexos, especialmente aqueles que envolvem incertezas.

A SMC baseia-se na geração de amostragens de números aleatórios para simular a variabilidade de um fenômeno dos quais não se têm conhecimento antecipadamente dos resultados relacionados (Raychaudhuri, 2008) e obter uma distribuição com resultados possíveis. Ao definir distribuições de probabilidade para variáveis-chave, os analistas podem realizar milhares de simulações para obter uma gama de resultados possíveis, permitindo a avaliação de riscos e a tomada de decisões informadas (Rubinstein e Kroese, 2017). Essa abordagem permite aos analistas explorarem o impacto de diferentes variáveis e parâmetros sobre um determinado sistema, e assim auxiliar na tomada de decisões (Romli e Harmin, 2015). Contudo a previsão está relacionada com a qualidade das distribuições de probabilidade utilizadas e o número de simulações realizadas. Além disso, a interpretação dos resultados requer cautela, uma vez que a análise de dados gerados aleatoriamente pode levar a conclusões equivocadas se não for feita de maneira adequada (Fishman, 1996).

Diante disso, a simulação de Monte Carlo utilizando por base uma distribuição probabilística, a simulação de Monte Carlo torna capaz a obtenção de diversos valores aleatórios que podem representar, por exemplo, o comportamento estocástico dos tempos de processamento de uma tarefa em uma máquina.

2.4 ALGORITMOS DE *MACHINE LEARNING*

De acordo Michie *et al.* (1995), ML compreende um conjunto de procedimentos computacionais automatizados de ordem lógica ou binária, que são capazes de aprender uma tarefa a partir de dados. Os algoritmos de ML podem ser classificados em três tipos de aprendizagem: supervisionada, não-supervisionada, e por reforço (Janiesch *et al.*, 2021). A seguir são apresentadas suas principais características:

- **aprendizagem supervisionada:** envolve a existência prévia de dados com uma classificação pré-estabelecida que irá atuar como o "supervisor" do conjunto de dados relativo ao objeto que se deseja classificar ou prever. Dado um conjunto de dados, realiza-se o treinamento baseado em modelos matemáticos e estatísticos, que buscam padrões nos dados de entrada. Pode ser dividida em duas partes: regressão, na qual é possível prever um valor numérico, e a técnica de classificação, que tende a prever uma classe de rotulação (Mrabet *et al.*, 2021). São exemplos de algoritmos de aprendizado supervisionado: árvore de decisão,*Support Vector Machine* (SVM), Naive Bayes, regressão linear, regressão logística e outros;

- **aprendizagem não supervisionada:** uma abordagem de aprendizado de máquina em que o algoritmo é treinado em um conjunto de dados não rotulado, ou seja, o algoritmo busca descobrir padrões, similaridades, estruturas ou relações intrínsecas nos dados sem a orientação do conjunto de rótulos previamente definidos (Bishop e Nasrabadi, 2006). Um dos principais objetivos do aprendizado não supervisionado é explorar a estrutura subjacente dos dados e extrair informações úteis que possam ser úteis para análises posteriores ou tomadas de decisão. Isso pode incluir a identificação de agrupamentos naturais de dados (*clustering*), a redução da dimensionalidade dos dados para visualização ou compressão, detecção de padrões anômalos ou outliers (detecção de anomalias) (Hastie *et al.*, 2009). São exemplos de algoritmos de aprendizagem não supervisionada: *K-nearest neighborhood*, *K-means*, Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*), entre outros.

- **aprendizagem por reforço**: consiste na adaptabilidade em cenários dinâmicos. Em outras palavras, trata-se de um paradigma de aprendizado de máquina no qual um agente interage com um ambiente dinâmico, realizando ações e recebendo uma resposta na forma de recompensas ou penalidades, com o objetivo de aprender a tomar decisões que maximizem a recompensa cumulativa ao longo do tempo (Sutton e Barto, 2018). Seguem quatro conceitos importantes para o contexto de aprendizagem por reforço: o primeiro, refere-se ao **ambiente** no qual o agente está inserido e pode ser modelado de diversas maneiras, desde jogos até um sistema de controle; o segundo envolve o **estado**, este configura-se como uma representação do ambiente em um determinado momento, sendo observada pelo agente e usada para tomar decisões; o terceiro, está relacionado a **ação** que sinaliza às possíveis ações que o agente pode realizar em cada estado do ambiente; e o quarto está ligado a **recompensa**, esta é transcrita como um sinal de retorno que o agente recebe após realizar uma ação em um determinado estado, indicando o quão boa foi a ação. Há diversos algoritmos que envolvem aprendizagem por reforço, dentre os quais estão: *State-Action-Reward-State-Action (SARSA)*, *Policy Gradient Methods* e o QL.

As técnicas de ML apresentam uma notável versatilidade. Isto se traduz, na aplicabilidade ao longo dos anos em diversas áreas do conhecimento, resultando em uma uma gama de funcionalidades largamente utilizadas, tais como: reconhecimento de fala, reconhecimento de imagens, sistemas de recomendação de músicas, filtragem de *spam* e vários outros exemplos.

2.4.1 Overfitting e Underfitting

Em estudos de ML explorar os conceitos de *overfitting* e *underfitting*, torna-se algo de grande relevância, haja vista que a ocorrência de ambos fenômenos pode comprometer a efetividade de um modelo preditivo. O *overfitting* ocorre quando o modelo de previsão escolhido, tende a levar em consideração todas as tendências, ruídos e estruturas que compõe o conjunto de dados de treinamento, ou seja, o modelo está sobreajustado aos dados (Larose e Larose, 2014). Quando isso acontece as métricas de desempenho apresentarão excelentes resultados. Desta forma, levando ao entendimento que o modelo é representativo para o caso abordado, o que não corresponde à realidade. Acontece que o modelo aprendeu muito bem os dados de treinamento, mas apresenta um desempenho ruim frente a dados novos, que não foram treinados, assim um modelo com *overffinting* tem sua capacidade de generalização comprometida (López *et al.*,2022). Em outras palavras o modelo não consegue realizar previsões adequadamente.

O fenômeno oposto ao *overfitting*, denomina-se *underfitting* que corresponde ao cenário, no qual o modelo preditivo é muito simples ou o conjunto de dados de treinamento não é capaz de representar o padrão geral dos dados para realização de previsões adequadas (López *et al.*,2022). Assim o modelo vai apresentar um desempenho ruim. A Figura 6 ilustra a diferença entre o *overfitting*, melhor ajuste e o *underfitting*.

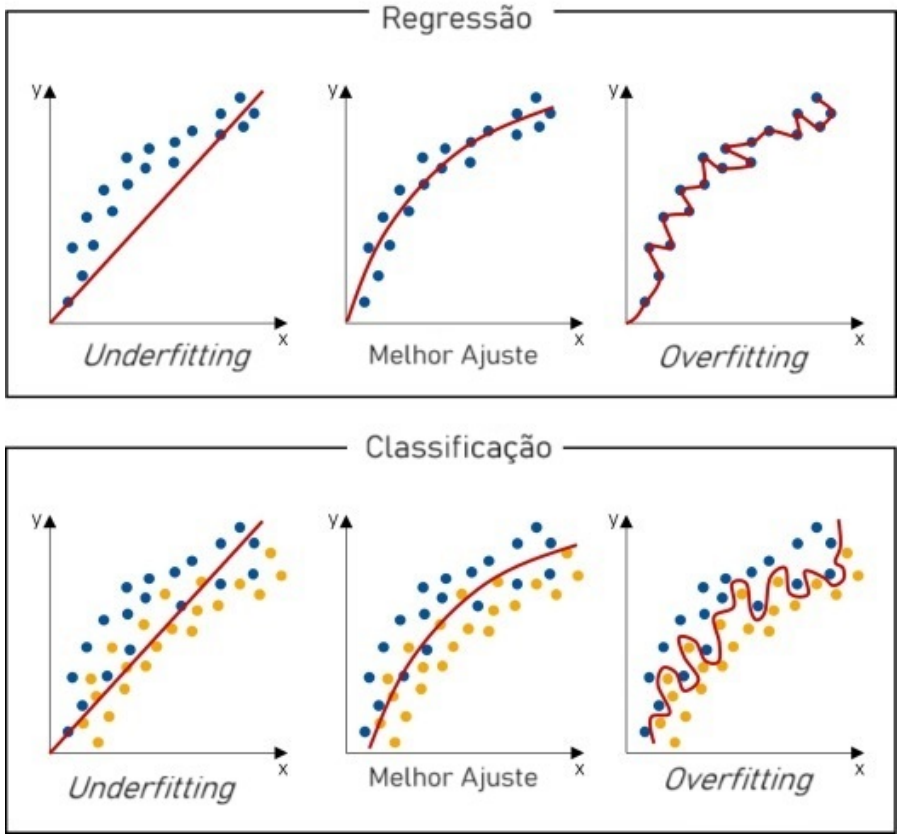
2.4.2 Regressão Linear Múltipla

A Regressão Linear é um modelo estatístico, utilizado de forma recorrente em análises preditivas. A ideia geral do modelo consiste em estabelecer uma relação linear entre uma variável dependente Y e uma variável independente X, com base na equação da reta (Yan e Su, 2009). O erro aleatório ρ não é decorrente do modelo de regressão linear, geralmente, assumido como uma variável normalmente distribuída com média igual a zero ($\mathbb{E}(\rho) = 0$) e variância $\text{Var}(\rho) = \sigma^2$ constante, logo $\rho \sim N(0, \sigma^2)$ (Doane e Seward, 2016; Yan e Su, 2009). A Figura 7 ilustra graficamente um ajuste por regressão linear simples.

$$y = \beta_1 X_1 + \beta_0 + \rho \tag{2.5}$$

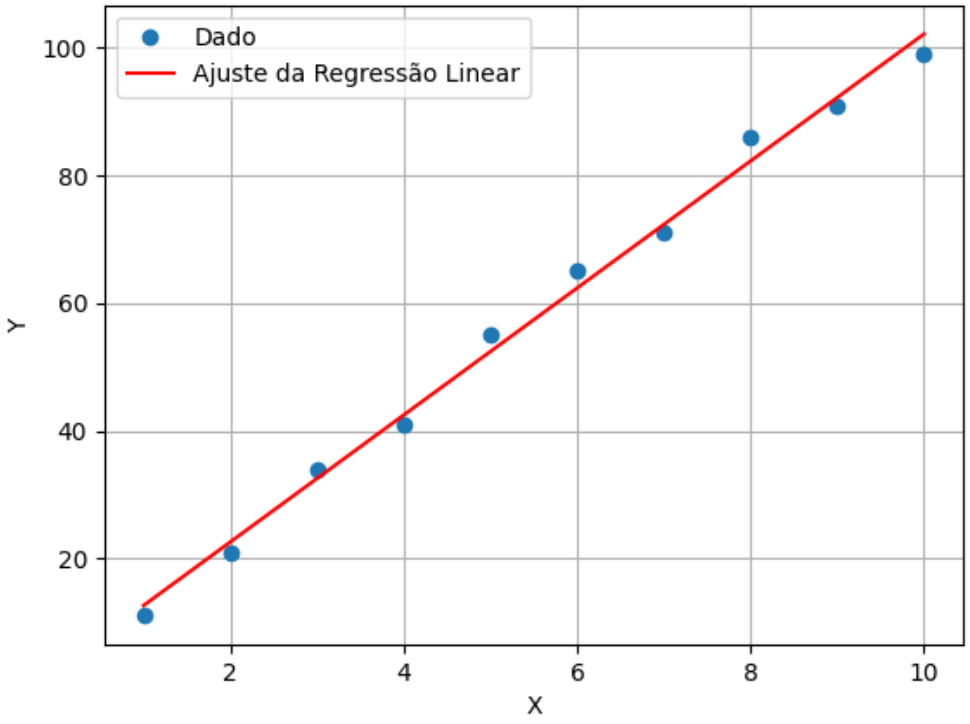
A Regressão linear múltipla é uma extensão de uma regressão linear simples, que envolve diversas variáveis independentes, denominadas variáveis preditoras e uma variável dependente, também chamada de variável predita ou explicada (Doane e Seward, 2016). A equação geral que representa

Figura 6 – Representação dos ajustes de *overfitting*, melhor ajuste e *underfitting*



Fonte: adaptado de Uhlig *et al.* (2023)

Figura 7 – Exemplo ilustrativo de uma Regressão Linear Simples



Fonte: Elaborado pelo autor (2025)

uma regressão linear múltipla é dada por:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_l X_l + \rho \tag{2.6}$$

y: variável dependente ou explicada

β_0: termo independente

$\beta_1, \beta_2, \beta_3, \ldots, \beta_l$: coeficientes do modelo de regressão

X:valor da variável preditora

l: índice da variável preditora

ρ : erro aleatório

2.4.3 Avaliação do Modelo de Regressão Linear

Após a realização do treinamento de qualquer algoritmo de ML, faz-se necessário avaliar a adequabilidade do modelo escolhido aos dados, visando interpretar se o modelo escolhido realmente é capaz de generalizar o comportamento dos dados. Vale salientar, que em muitos casos utilizar mais de um tipo de métrica pode trazer uma visão mais completa. No caso, da regressão linear, uma das formas de avalia-la na compreensão do ajuste realizado, utiliza-se o coeficiente de determinação (R^2). Esta medida estatística é capaz de medir a proporção do quanto a variável dependente pode explicar os preditores (Martin, 2021). A Equação (2.7) representa o cálculo do coeficiente de determinação, a Equação (2.8) demonstra o cálculo da Soma Residual dos Quadrados, a Equação (2.9) apresenta o cálculo da Soma dos Quadrados da Regressão, a Equação (2.10) a Soma Total dos Quadrados.

$$R^2 = 1 - \frac{SS_{\text{regressão}}}{SS_{\text{total}}} \tag{2.7}$$

Onde,

$$SS_{\text{residual}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.8}$$

$$SS_{\text{regressão}} = \sum_{i=1}^{N} (y_i - \bar{y})^2 \tag{2.9}$$

$$SS_{\text{total}} = SS_{\text{regressão}} + SS_{\text{residual}} \tag{2.10}$$

R^2: coeficiente de determinação

SS_{residual}: soma dos quadrados dos erros (residuais)

$SS_{\text{regressão}}$: soma dos quadrados da regressão

SS_{total}: soma total dos quadrados

y_i: valor observado

$\hat{y}_i$: valor previsto

$\bar{y}$: média da variável explicada

Na regressão linear, o coeficiente de determinação varia entre 0 e 1. Quanto mais próximo de zero, significa que as variáveis independentes não são capazes de explicar a variável dependente, e quanto mais próximo de um denota que as variáveis preditoras são capazes de explicar a variável dependente. Por exemplo, se o valor de $R^2 = 0,87$, pode-se entender que as variáveis independentes do modelo são capazes de explicar 87% dos resultados da variável dependente.

Contudo, algumas métricas referente ao erro associado ao modelo de regressão, tornam-se muito útil no entendimento da eficácia preditiva. Neste caso, pode-se utilizar o Erro Médio Absoluto (EMA) e a Raiz do Erro Médio Quadrático (REMQ), respectivamente Equações (2.11) e (2.12).

$$\text{EMA} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.11}$$

$$\text{REMQ} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.12}$$

2.4.4 Q-LEARNING

O *Q-learning* (QL) é um algoritmo que envolve aprendizado por reforço, proposto por Watkins (1989). Este visa fornecer aos agentes capacidade de aprender a tomar decisões de forma otimizada a partir de domínios markovianos, testando sequências de ações sem implicar na construção de mapas de domínios (Watkins e Dayan, 1992). Nesse contexto, os Processos de Decisão de Markov (*MDPs*, do inglês *Markov Decision Processes*) são de grande relevância no entendimento do funcionamento do *Q-learning*, os *MDPs* referem-se a uma modelagem matemática para representar problemas de decisão sequencial e a tomada de decisão sob condições de incerteza (Sigaud e Buffet, 2010).

Nesse caso, utilizando a Equação de Bellman (Watkins1989; He *et al.*, 2022; Guo *et al.*, 2024):

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{T} \gamma^{k} r_{t+k+1} \mid s_t = s\right] \tag{2.13}$$

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{T} \gamma^{k} r_{t+k+1} \mid s_t = s\right] \tag{2.14}$$

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[r_{t+1} + \gamma \sum_{k=0}^{T} \gamma^{k} r_{t+k+2} \mid s_t = s\right] \tag{2.15}$$

$$V^{\pi}(s) = \sum_{s' \in S} T_a(s,s') \left[R(s,a,s') + \gamma V^{\pi}(s')\right] \tag{2.16}$$

$$V^{*}(s) = \max_{a \in A} \sum_{s' \in S} T(s,a,s') \left[R(s,a,s') + \gamma V^{*}(s')\right] \tag{2.17}$$

Onde, $V^{\pi}(s)$ indica o valor da função estado para uma política π que fornece o total esperado ao longo de um período, t = 1, 2, 3 ... T. O valor da recompensa de tomar uma ação a_{t+k} estando no estado s_{t+k} é representado por r_{t+1+k}, enquanto γ^{k} é o fator de desconto, $T(s,a,s')$ representa a probabilidade de transição de tomar a ação a no estado s e resultando no estado s, o mesmo raciocínio se aplica a recompensa $R(s,a,s')$.

Resumidamente, o processo pode ser definido pela tupla $(S,A,A_{.},P,R)$, onde S é o espaço de estados, A representa o conjunto de ações, $A_t \subseteq A$ é o conjunto de ações possíveis no estado $S_t \in S$, $P(S_t,a)(S_{t+1})$ é a probabilidade de transição do estado $S_t \in S$ para o estado $S_{t+1} \in S$, considerando a ação $a \in A$ é tomada, e $R(S_t,a)$ é a recompensa obtida quando no estado $S_t \in S$ é tomada a ação $a \in A_t$. Onde θ é a taxa de aprendizagem e o γ a taxa de desconto. Deste modo a equação do *Q-learning* pode ser resumida em (Sigaud e Buffet, 2010) :

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \theta \left[R_{t+1} + \gamma \max_{a} Q(S_{t+1},a) - Q(S_t,A_t)\right] \tag{2.18}$$

Com a Equação (2.18) é possível atualizar a matriz de estado-ação do algoritmo *Q-learning* com os valores da recompensa e a cada episódio (repetição para o treinamento) o algoritmo pode tomar melhores decisões ou ainda pode-se adotar estratégias como a exploração-explotação.

A explotação envolve maximizar as recompensas com base nas informações já adquiridas. Nesse caso, o agente opta por ações que, de acordo com seu conhecimento atual, têm maior probabilidade de resultar em recompensas positivas. Do outro lado, a exploração refere-se à tentativa de novas ações que o agente ainda não testou ou que têm informações limitadas sobre suas consequências. Partindo do pressuposto que descobrir novas informações que podem levar a melhores resultados a longo prazo. O processo de exploração ajuda a garantir que o agente não perca oportunidades de aprender e melhorar seu desempenho (Bertsekas e Tsitsiklis, 1996).

Diferentes algoritmos e técnicas são projetados para lidar com esse *trade-off* de maneiras variadas. Um dos métodos utilizado para balancear a exploração e explotação de forma controlada seria a estratégia como $\varepsilon - greedy$ (Gimelfarb *et al.*, 2020). Nesta estratégia, o agente pode escolher uma ação aleatoriamente com uma pequena probabilidade ε, permitindo a exploração, enquanto na maioria das vezes escolhe ações as melhores ações com base na explotação.

3 REVISÃO DA LITERATURA

Este capítulo apresenta a revisão da literatura relacionada ao tema desta pesquisa, a qual baseou-se numa coletânea de artigos científicos, visando entender o progresso do desenvolvimento dos métodos associados a resolução do PFSP. Ressaltando os algoritmos heurísticos, meta-heurísticos, técnicas híbridas, simheurística, e por fim, enfatizar as aplicações com ML no contexto do *flow shop*.

Apesar das limitações dos algoritmos exatos para resolução de grandes instâncias, o trabalho pioneiro na resolução do PFSP se deu de maneira exata, onde um algoritmo proposto por Johnson (1954) obteve a solução ótima para um PFSP com duas máquinas. No decorrer do tempo, outras pesquisas aplicaram métodos exatos para a resolução de problemas *flow shop*, por exemplo, o algoritmo *branch-and-bound* foi empregado para resolver problemas com arranjos de duas máquinas por Ignall e Schrage (1965) que utilizaram o método para resolução de 09 tarefas com intuito de minimizar *makespan*, assim como Velde (1990) propôs uma abordagem para resolução de um problema *flow shop* com relaxação lagrangiana nas restrições de precedência e nos limites inferiores, também visando a minimização do *makespan*. Inobstante a problemática, Brown e Lomnicki (1966) desenvolveram uma proposição para resolução do PFSP em um arranjo com três máquinas. Potts (1980), usou em sua abordagem um algoritmo *branch-and-bound* fixando a primeira e a última tarefa e incluindo regras de dominância, os experimentos computacionais realizados evidenciaram a superioridade desse algoritmo em comparação aos métodos que haviam sido publicados até aquele momento. Na atualidade, Liu e Urgo (2024) propuseram uma abordagem de *branch-and-bound* para resolver o problema PFSP em duas máquinas e as peças processadas em lotes, objetivando a minimização do *value-at-risk* do *makespan* com a intenção de auxiliar processo de tomada de decisão para mitigar impactos cenários extremos.

No campo das heurísticas, há uma diversidade de métodos que se propõem a resolver o PFSP, Campbell *et al.* (1970) desenvolveram uma heurística construtiva baseada no algoritmo de Jhonson (Johnson, 1954) para obter soluções aproximadas de um sequenciamento de muitas tarefas, essa heurística ficou conhecida como CDS. A heurística de Palmer (Palmer, 1965), cuja a ideia básica está ancorada no cálculo de um índice de "declive"para cada tarefa, em seguida as tarefas são ordenadas com base nesse índice em ordem não crescente. A Heurística construtiva mais conhecida aplicada ao PFSP é a heurística NEH, mencionada no Capítulo 2, proposto por Nawaz *et al.* (1983), esta heurística parte do ordenamento das tarefas com base no tempo de processamento mais longo, seleciona as duas primeiras tarefas resultantes do ordenamento da lista e em seguida, verifica-se elemento a elemento da lista inicial a melhor posição para inserção de uma tarefa, de forma a minimizar o $C_{\text{máx}}$. Ainda no

Capítulo 2 foi abordado o método desenvolvido por Taillard (1990), capaz de executar a heurística NEH de forma eficiente, conhecida como NEHT.

Há também uma variedade de abordagens meta-heurísticas utilizadas para resolução do PFSP, Osman e Potts (1989), aplicaram a meta-heurística SA, com o objetivo de minimizar o *makespan*. Assim como, Murata *et al.* (1996) implementaram algoritmos genéticos associados a procedimentos de busca local, objetivando encontrar o sequenciamento que minimiza o *makespan*. Nowicki e Smutnicki (1996) desenvolveram um algoritmo que utiliza a meta-heurística busca tabu com uma especificidade na definição da vizinhança que emprega a ideia de blocos de tarefas. Stützle (1998) aplicou a meta-heurística ILS (*Iterated Local Search*) com modificações na implementação, visando resolver o PFSP. Ríos-Mercado e Bard (1998) apresentaram dois métodos para uma variação FSP que corresponde a consideração de tempos de *setup* dependentes do sequenciamento, na ocasião o primeiro método abordado foi uma extensão da heurística NEHT e o segundo baseou-se na meta-heurística GRASP. Ramanan *et al.* (2014) implementaram um algoritmo que utiliza por base meta-heurística PSO (*Particle Swarm Optimization*), com a intenção de minimizar o *makespan*, na abordagem implementada o algoritmo NEH é utilizado para construir uma solução, em seguida o PSO orienta a busca das soluções e em logo depois, a meta-heurística de busca de vizinhança variável (VNS) é aplicada para auxiliar o PSO na busca de uma solução global.

Além do critério $C_{\text{máx}}$ a literatura retratam outras métricas, Xiao *et al.* (2012) demonstram uma perspectiva diferente, ao abordarem o problema do PFSP com aceitação de pedidos e atrasos ponderados (PFSP-OAWT, do inglês *Permutation Flowshop Scheduling Problem with Order Acceptance and Weighted Tardiness*). Neste estudo, o objetivo principal compreendeu a maximização do lucro líquido total com penalidades de atrasos ponderados, desenvolvendo um algoritmo meta-heurístico *simulated annealing* baseado em otimização parcial. Liu *et al.* (2018) desenvolveram um trabalho considerando como critério a minimização do consumo total de energia de máquinas ociosas, encontrando solução ótima para duas máquinas através da relaxação do algoritmo de Jhonson. Assim como, Schaller e Valente (2019) abordaram o problema do PFSP usando como critério a minimização a totalidade do adiantamento e atraso com tempo ocioso não forçado, utilizando conjuntos de heurísticas de despacho e heurísticas de expedição. Uslu *et al.* (2019) apresentaram um algoritmo de Busca Local Auto-adaptativa Modificada (MSALS do inglês, *Modified Self-Adaptive Local Search*) de abordagem biobjetiva, constituindo-se em fins a minimização do *makespan* e do tempo total de fluxo, no contexto do *flowshop* permutacional.

Uma metodologia proposta por Jaradat *et al.* (2016), estabelece o uso de uma estrutura de memória denominada *pool* que armazena soluções de alta qualidade em meta-heurísiticas de busca

populacional, testada em problemas do caixeiro viajante, roteamento de veículos e problema da mochila.

Na perspectiva da junção das técnicas de simulação com métodos heurísticos e meta-heurísticos, Gonzalez-Neira *et al.* (2021) propuseram uma abordagem simheurísitica que compreende uma simulação de Monte Carlo em uma meta-heurística GRASP hibridizada com a PAES (*Pareto Archived Evolution Strategy*), vislumbrando uma otimização multiobjetivo. Ademais, integrando a metodologia AHP *(Analytic Hierarchy Process)* a simheurística com o intuito de avaliar as soluções de pareto sob diferentes matrizes de pesos no que se refere aos critérios selecionados. As análises estatísticas baseadas na ANOVA, demonstraram resultados estatisticamente significantes para funções objetivo escolhidas: antecipação/atraso esperado, desvio-padrão da antecipação/atraso, importância esperada do trabalho para o cliente e importância esperada do produto. Outra técnica de hibridização foi proposta por Istokovic *et al.* (2020), baseando-se em um modelo de simulação-otimização que une simulação de eventos discretos com a meta-heurística algoritmo genético na resolução do problema de sequenciamento de lotes em um *flow shop* híbrido, comumente utilizado em média e larga produção de produtos tecnológicos de alta complexidade. O estudo conseguiu agrupar três produtos tecnologicamente semelhantes em um único processo de produção, nos custos de produção e prazo ótimo, a metodologia abordada encontrou o resultado em um tempo computacional de 59 minutos.

As abordagens de inserção de incerteza estocástica, permeiam outras técnicas como é o caso de Gourgand *et al.* (2003) que desenvolveram modelos baseados na heurística CDS e com recozimento simulado em ambiente PFSP com *buffers* ilimitados, usando a distribuição exponencial para gerar os tempos de processamento estocásticos em cada máquina, além de recorrer a um modelo apoiado nas cadeias de Markov para calcular o *makespan* esperado e utilizando simulações de eventos discretos para avalia-lo. Juan *et al.* (2014) desenvolveu um método simheurístico para o PFSP com inserção de incerteza, capaz de realizar previsão do *makespan* esperado com base na SMC e na meta-heurística ILS. Villarinho *et al.* (2021) realizaram uma análise baseada no *flow shop* permutacional com datas de entregas e *payoffs* cumulativos em condições de incerteza, e tendo como objetivo maximizar o *payoff* esperado, estabelecendo a permutação das tarefas para isso, propondo uma heurística aleatória tendenciosa aliada a um método meta-heurístico e simheurístico. Amirghasemi (2021) apresentou um algoritmo estocástico baseado em decomposição de grande vizinhança em uma meta-heurística de busca de vizinhança variável (VNS), voltado para resolução do PFSP. Castaneda *et al.* (2022) que implementaram uma simheurística Fuzzy para o problema do *flow shop* permutacional usando tempos de processamento estocásticos, objetivando a minimização do *makespan* esperado. Rodriguez-Espinosa *et al.* (2024) desenvolveram um método simheurístico hibridizado que compreende o algoritmo NSGA-

II com a simulação de Monte Carlo com falhas estocásticas nas máquinas, considerando múltiplos objetivos como: *makespan*, tarefas em atrasos e o desvio-padrão das tarefas atrasadas para o contexto do *flowshop* flexível estocástico com quebra de máquinas.

A temática acerca da aplicação de algoritmos ML em problemas de FSP e PFSP ainda é incipiente no meio acadêmico. Contudo, há alguns trabalhos que se destacam com o uso das técnicas de aprendizado de máquina. Azadeh *et al.* (2015) desenvolveram um método que uni redes neurais artificiais a simulação, com objetivo de minimizar o custo total, no contexto de uma abordagem multi-objetivo em ambiente *flow shop* de máquinas paralelas idênticas. Por exemplo, Azab *et al.* (2021) empregaram modelos de *Machine Learning* no contexto de manutenção preditiva em situações envolvendo programação da produção dinâmica. Zhou *et al.* (2023) desenvolveram um trabalho utilizando aprendizagem por reforço profundo com intuito de minimizar o *makespan*. Outro trabalho destacado foi o desenvolvido por Müller *et al.* (2024) que partiram de uma abordagem baseada no algoritmo de aprendizagem por reforço Otimização de Política Proximal (PPO, do inglês *Proximal Policy Optimization*) para resolver o PFSP, no contexto da produção de eletrodomésticos.

Alguns trabalhos utilizaram a junção das técnicas de ML com métodos heurísticos e meta-heurísticos com intuito de obter soluções de mais qualidade, Ainda, Ren *et al.* (2021) utilizaram um modelo de aprendizado por reforço com multiagentes associado a um algoritmo NASH propuseram o algoritmo NASH Q-Learning, aplicado ao contexto do *flowshop* permutacional distribuído. He *et al.* (2022), desenvolveram uma metodologia baseada em aprendizagem por reforço, mais especificamente utilizando o algoritmo *q-learning*. Na ocasião, foi proposto um algoritmo que combina a heurística NEH com o QL com alterações na atualização da matriz Q para melhorar o algoritmo otimizador inicial, onde os resultados das simulações realizadas demonstraram a eficácia do método para as instâncias testadas, tal algoritmo foi chamado de *Improvement Q-Learning* (IQL). Um trabalho similar foi desenvolvido por Guo *et al.* (2024) que instrumentalizaram o algoritmo QL e o uniram a heurística NEH, o algoritmo implementado superou o QL padrão, com base no critério de minimização do *makespan*. Li *et al.* (2023) propuseram a melhoria de um algoritmo heurístico de colônia artificial de abelhas (ABC), utilizando o *Q-learning* para escolher as melhores estruturas de vizinhança nas iterações, objetivando a minimização do *makespan*. Karimi-Mamaghan *et al.* (2023) desenvolveram um algoritmo que usa o *q-learning* para selecionar o operador de perturbação apropriado, incorporando a ferramenta em uma meta-heurística. Ying *et al.* (2024) propuseram um algoritmo que foi chamado de *Reinforcement-Learning-Based-*α*-List-Iterated-Greedy* (RAIG), mecaniza uma *N-List* como solução inicial e em seguida a fase refinamento da solução é aprimorado com *reinforcement learning* e a α*-list*.

Outros projetos abordaram métodos mais complexos como o emprego de técnicas de

aprendizado profundo (*Deep Learning*), Pan *et al.* (2023) propuseram a criação de uma rede neural de aprendizado profunda que visa minimizar o tempo máximo de conclusão, neste trabalho destaca-se o uso de um método não rotulado que não depende da criação de dados rotulados de alta qualidade. Lu *et al.* (2024) com os objetivos de minimizar o *makespan* e o consumo total de energia, uniu o aprendizado profundo a um algoritmo evolutivo de decomposição, denominado de GDRL-MOEA/D,em seguida compara com diversos algoritmos bioinspirados e métodos clássicos de otimização multi-objetivo MOEA/D (do inglês, *Multiobjective Evolutionary Algorithm Based on Decomposition*) e NSGA-II (do inglês, *Nondominated Sorting Genetic Algorithm II*), ainda segundo o estudo os resultados do algoritmo GDRL-MOEA/D superaram os demais métodos em termos de qualidade de solução. Marchesano *et al.* (2021) desenvolveram um método que utiliza *Deep Q-Network* (DQN), qual está inserido no contexto do *Deep Reinforcement Learning*, objetiva conceber uma política de sequenciamento auto-otimizante. Com um conjunto de regras de despacho previamente conhecidas para cada fila de máquinas, entre as quais a melhor é selecionada dinamicamente, considerando o estado do sistema.

Ainda, há trabalhos relacionados ao tema com aprendizado supervisionado Benda *et al.* (2019) propuseram um estudo baseado em *machine learning* para o *flow shop* com recursos alternativos tempos de *setup* dependentes de sequência e bloqueios, objetivando minimizar o *makespan* e o tempo total de atraso. Na ocasião a proposição se destina a gerar uma regra de distribuição em base do método de Árvores de decisão dependentes de soluções de alta qualidade, por sua vez obtidas por meio de uma programação de restrições. Ainda o estudo utilizou,*random forest* para tentar lidar com *overffiting*, Souza *et al.* (2024) desenvolveram algoritmos de regressão com regularização para estimativa do *makespan* esperado, mais especificamente hibridizando a meta-heurística *multi-start* com os modelos de regressão com regularização L1 e L2 e utilizando a SMC para simular a estocasticidade dos tempos de processamento de cada tarefa em cada máquina.

No contexto do *flow shop* de máquinas paralelas,Yamashiro e Nonaka (2021) desenvolveram um método que realiza a estimação dos tempos de processamento com dados reais, através de métodos de ML. O estudo chegou a conclusão que os algoritmos *Light Gradient-Boosted Machine* e Regressão Rígida apresentaram os melhores resultados em termos de REMQ e do Erro Médio Absoluto Percentual. A solução implementada resultou em uma redução de 30 % no valor do *makespan*.

A literatura correlacionada ao tema PFSP com abordagens que envolvem técnicas de ML não é tão vasta. No entanto, nota-se que há uma tendência de hibridização de técnicas, com o propósito de melhorar os resultados obtidos e consequentemente apresentar uma solução mais próxima do ideal capaz de representar adequadamente a realidade. Nesta conjuntura, o trabalho desta pesquisa visa propor e comparar métodos híbridos que estabeleçam o uso concomitante de simulação, técnicas de

ML, heurísticas e meta-heurísticas, objetivando realizar quanto ao *makespan* esperado em ambiente PFSP com considerações estocásticas, visando aumentar a precisão das previsões, estabelecendo melhores sequenciamentos. Nesse contexto, esta dissertação propõe alguns métodos que vislumbram o fim supramencionado, no que se refere a simulação foi proposto o método de Simulação Com Matrizes. No que se refere às técnicas híbridas de heurísticas com modelos supervisionados, propuseram-se os métodos: *Multi-start* com NEHT e Regressão Linear Múltipla (MSNEHT-RLM-POOL) e GRASP com Regressão Linear Múltipla (GRASP-RLM-POOL); no contexto de aprendizado por reforço, foram construídos o *Q-Learning NEH - Simulation* (QLNEH-S), Improvement Q-Learning - Simulation (IQL-S) e, os respectivos híbridos *Multi-start Q-Learning NEH - Simulation* (MSQLNEH-S) e *Multi-start Improvent Q-learing - simulation* (MSIQLS-S); também foram propostos métodos simheuísticos equivalentes, chamados de *Multi-start* NEHT - Simheurística (MSNEHT-SIMH) e o GRASP - Simheurística (GRASP-SIMH). Todos estes métodos mencionados agregam o modelo proposto de Simulação com Matrizes.

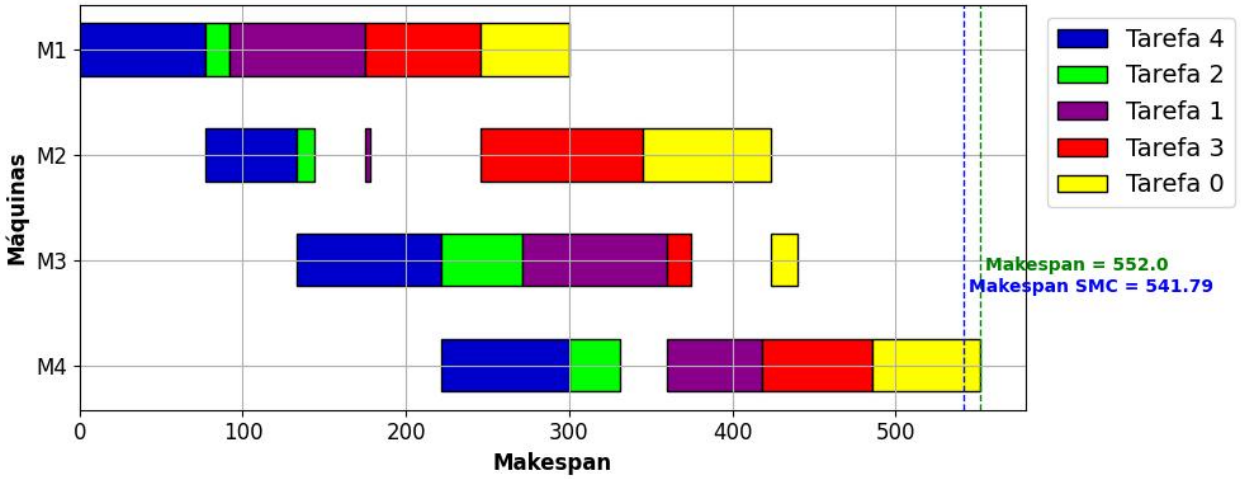
4 DESCRIÇÃO DO PROBLEMA

Neste capítulo, é apresentada uma visão geral acerca do Problema do Sequenciamento *Flow Shop* Permutacional, descrevendo os principais desafios e a importância em buscar meios para resolução desse problema.

O FSP é um problema clássico na programação da produção, comumente encontrado em ambientes industriais, onde diferentes etapas de produção precisam ser realizadas em máquinas específicas. Nesse contexto, a gestão adequada do fluxo de produção, busca encontrar o sequenciamento ideal das tarefas em um determinado conjunto de máquinas. Deste modo, um conjunto de n tarefas devem ser processadas em m máquinas, denotando uma operação da tarefa i na máquina j. No caso do PFSP, este processamento deve ocorrer na mesma ordem em todas as máquinas.

A solução desse problema perpassa pela complexidade em determinar possíveis sequências na ordem ($n!$). A sequência de tarefas nas máquinas deve ser capaz de minimizar ou maximizar uma métrica específica estabelecida. Conforme abordado anteriormente, o *makespan* é uma das métricas mais usuais para este tipo de problema, cujo objetivo principal corresponde a minimizá-lo. A Figura 8 ilustra um exemplo de arranjo do PFSP, considerando um ambiente que utiliza n = 5 tarefas e m = 4 máquinas, destacando-se, os *makespans* calculados através dos tempos médios e de uma SMC, dada a sequência de tarefas [4 , 2, 1, 3, 0]. A Figura 9 ilustra a mesma configuração, porém os blocos representam os tempos decorrentes de uma SMC.

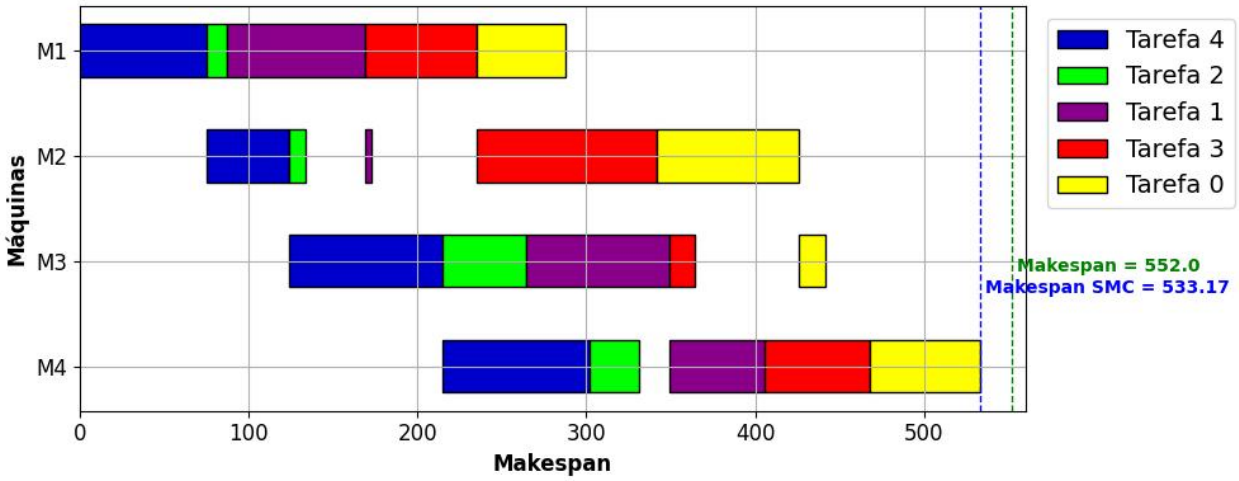
Figura 8 – Gráfico de Gantt ilustrando o PFSP para os tempos médios



Fonte: Elaborado pelo autor (2025)

Conforme explanado anteriormente, o PFSP possui um caráter combinatório, pois há muitas combinações possíveis para o sequenciamento das tarefas nas máquinas. Assim, o custo computacional demandado para testar todas as combinações possíveis, tem como causalidade a inadequação relativa à

Figura 9 – Gráfico de Gantt ilustrando o PFSP para os tempos da SMC



Fonte: Elaborado pelo autor (2025)

utilização de métodos capazes de encontrar a solução ótima, ou seja, a melhor sequência existente que possa minimizar o *makespan*. Por exemplo, uma vez que o tempo computacional exigido mostra-se inviável para resolução do problema de acordo com a sua complexidade.

Um aspecto importante a ser considerado, refere-se à inserção de estocasticidade nos tempos de processamento das tarefas em PFSP, pois estes não são tempos determinísticos em aplicações reais. Assim, visando obter resultados mais satisfatórios e próximos da realidade, tal estudo propõe a aplicação de métodos de otimização (*i.e.*, algoritmos heurísticos e meta-heurísticos), capazes de determinar boas sequências para o ordenamento das tarefas nas máquinas. Combinado com o uso de simulações computacionais e a aplicação de técnicas de ML com a intenção de encontrar soluções de qualidade para problemas desse tipo.

4.1 PRESSUPOSTOS DO PROBLEMA

O PFSP segue um conjunto de pressupostos para que seja possível firmar os arcabouços estruturais dos experimentos e das análises decorrentes. Os pressupostos adotados (Öztop *et al.*, 2020; He *et al.*, 2022) são:

a. não há restrições de precedência entre as tarefas;
b. a execução em cada máquina são independentes;
c. não são considerados os tempos de configuração das máquinas;
d. as máquinas processam as tarefas sempre na mesma sequência;
e. não há preempção ou qualquer tipo de interrupção de tarefas na máquina.

4.2 *MAKESPAN*

Uma modelagem matemática genérica que pode ser utilizada para o cálculo do *makespan* ($C_{\text{máx}}$), baseia-se nos Tempos de Início de uma tarefa i em uma máquina j (TI_{ij}), cujo cálculo é descrito nas Equações (4.1) e (4.2). Enquanto a Equação (4.3) descreve o cálculo do *makespan* esperado, baseado na SMC.

$$TI_{ij} = \begin{cases} 0, & \text{se } i = 1 \text{ e } j = 1 \\ TI_{i-1,j} + TPE_{i-1,j}, & \text{se } i > 1 \text{ e } j = 1 \\ TI_{i,j-1} + TPE_{i,j-1}, & \text{se } i = 1 \text{ e } j > 1 \\ \text{máx}(TI_{i-1,j} + TPE_{i-1,j}, TI_{i,j-1} + TPE_{i,j-1}), & \text{c. c.} \end{cases} \tag{4.1}$$

$$C_{\text{máx}} = TI_{n,m} + TPE_{n,m} \tag{4.2}$$

$$\mathbb{E}[C_{\text{máx}}] = \frac{\sum_{i=1}^{n_{\text{iterações}}} C_{\text{máx}_i}}{n_{\text{iterações}}} \tag{4.3}$$

5 METODOLOGIA

Neste capítulo, são apresentados a metodologia do trabalho dividida em: proposição do método de simulação matricial (Simulação Com Matrizes), meta-heurísticas híbrida com o modelo de aprendizagem supervisionada, modelos de aprendizagem por reforço envolvendo questões estocásticas e meta-heurísticas híbridas com esses métodos. Por fim, apresenta-se dois modelos desenvolvidos para fins comparativos que estima o *makespan* apenas considerando os tempos médios.

5.1 SIMULAÇÃO ESTOCÁSTICA

Neste trabalho, a simulação dos Tempos de Processamento Estocástico (P_{ij}) de cada tarefa i em cada máquina j segue a distribuição Log-normal, *i. e.*, $P_{ij} \sim \text{LogN}(\mu_{ij}, \sigma_{ij})$, tal qual a abordagem de estocasticidade apresentada em Juan *et al.* (2014), cujos parâmetros são calculados com base nas equações abaixo:

$$\text{Var}[P_{ij}] = \mathbb{E}[P_{ij}] \cdot \kappa \qquad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m \tag{5.1}$$

$$\mu_{ij} = \ln\left(\mathbb{E}[P_{ij}]\right) - \frac{1}{2}\ln\left(1 + \frac{\text{Var}[P_{ij}]}{\mathbb{E}[P_{ij}]^2}\right) \qquad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m \tag{5.2}$$

$$\sigma_{ij} = \left|\sqrt{\ln\left(1 + \frac{\text{Var}[P_{ij}]}{\mathbb{E}[P_{ij}]^2}\right)}\right| \qquad i = 1,2,\ldots,n;\ j = 1,2,\ldots,m \tag{5.3}$$

Onde,

P_{ij}: Tempo de Processamento Estocástico da tarefa i na máquina j

$\mathbb{E}[P_{ij}]$: Tempo de Processamento Médio da Tarefa i na Máquina j

$\text{Var}[P_{ij}]$: Variância do Tempo de Processamento da Tarefa i na Máquina j

σ_{ij}: Desvio-padrão

μ_{ij}: Média

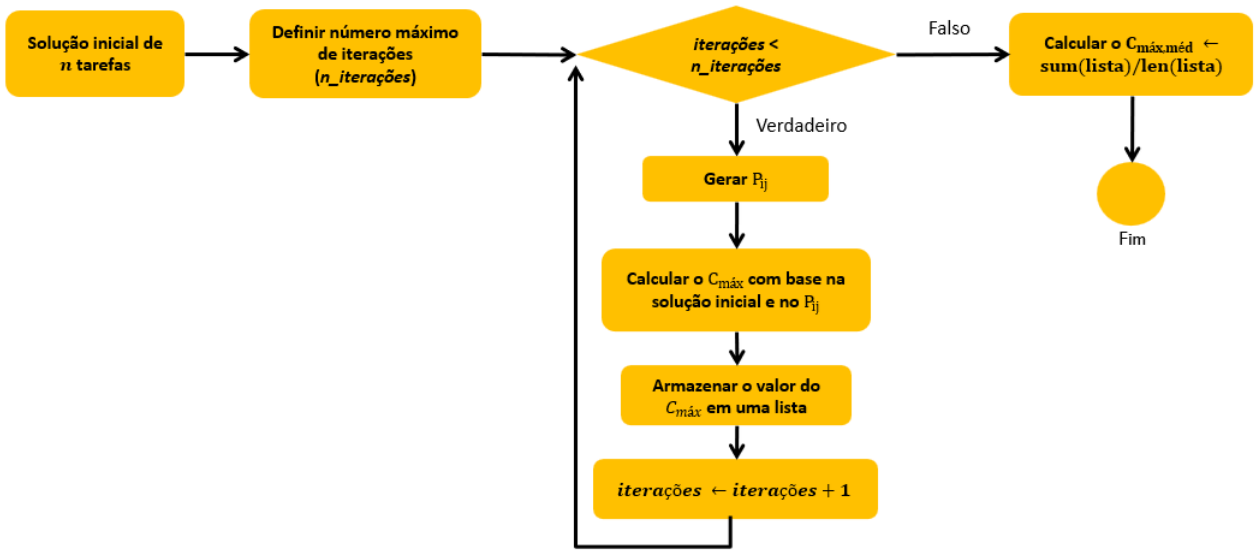
κ: Coeficiente para Cálculo da Variância

A Equação (5.1), calcula a variância associada ao tempo de processamento médio, enquanto as Equações (5.2) e (5.3) estão relacionadas ao cálculo da média e do desvio-padrão dos tempos de processamento de cada tarefa i em cada máquina j para representação dos tempos estocásticos. Os Tempos de Processamento Médios ($\mathbb{E}[P_{ij}]$) utilizados neste trabalho correspondem aos valores

disponíveis nas instâncias de Taillard (Taillard, 1993) para o FSP médio. Os $\mathbb{E}[P_{ij}]$ são armazenados na Matriz de Tempos de Processamento Médio (MTPM).

A forma de realização da SMC pode ser obtida por meio da média direta entre os valores dada uma geração de dados. Este ponto é entendido, no contexto deste trabalho como um método clássico de gerar a simulação de Monte Carlo para estimar o *makespan*. Para fins, este modo que efetiva a SMC está sendo identificado como Simulação Clássica, haja vista que um modo comum de usar a SMC para calcular o *makespan* médio ($C_{\text{máx, médio}}$). O Algoritmo 5 demonstra o pseudocódigo do procedimento de simulação clássica. Enquanto, a Figura 10 apresenta o fluxograma ilustrativo da Simulação Clássica.

Figura 10 – Fluxograma ilustrativo da Simulação Clássica



Fonte: Adaptado de JUAN *et al.* (2014)

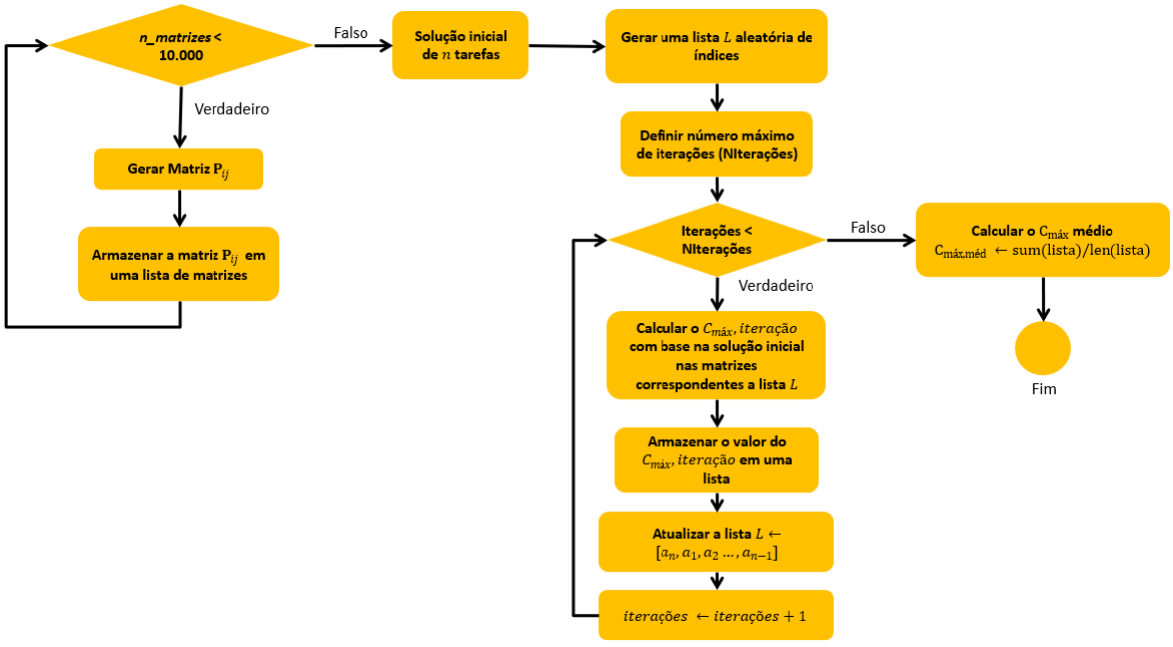
Algoritmo 5: SimulaçãoClássica(n, m, κ, MTPM, n_iterações)

```
sol_inicial ← solução de n tarefas
máx_iteração ← n_iterações
lista_makespans ← []
iter ← 0
C_máx, médio ← 0
while iter do
    P_ij ← SimulandoTemposAleatórios(sol_inicial, n, m, κ,P_ij)
    C_máx,it ← CalculandoMakespanIteração(sol_inicial, n, m, P_ij)
    lista_makespans.append(C_máx,it)
    iter ← iter + 1
end while
C_máx,médio ← sum(lista_makespans)/len(lista_makespans)
return C_máx,médio
```

5.2 MÉTODO MATRICIAL PROPOSTO PARA CALCULAR O *MAKESPAN* ESPERADO SIMULADO, NO CONTEXTO DO *FLOW SHOP* PERMUTACIONAL

A forma proposta para calcular a simulação estocástica para estimação do *makespan* esperado, envolve a geração de 10.000 matrizes compostas pelos tempos de processamento estocástico de cada tarefa i em cada máquina j, também admitindo $P_{ij} \sim \text{LogN}(\mu_{ij}, \sigma_{ij})$, e cujos parâmetros são calculados com base nas Equações (5.1), (5.2) e (5.3), a partir da MTPM. Cada matriz com tempos simulados (estocásticos) é armazenada em um lista, e recebem um índice associado. Posteriormente, gera-se uma lista L de índices aleatórios, sem repetição. Em seguida, o usuário deve definir o número de iterações (n_iterações) desejadas, logo o procedimento principal é executado, resgatando, através dos índices da lista L, os valores do P_{ij} para cada tarefa dada uma solução inicial e calculando um *makespan* de cada iteração ($C_{\text{máx},it}$) e o armazenando em uma lista (lista_makespans), ressaltando que o tempo de processamento de cada tarefa será calculado com base no valor simulado de cada matriz correspondente a lista L, pois a cada iteração há um deslocamento à direita de cada índice constituindo a atualização de L com $L = [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$, garantindo que o valor de cada iteração deve ser proviniente de uma matriz diferente. Ao final do procedimento, calcula-se o $C_{\text{máx, médio}}$, que corresponde a média dos valores calculados nas iterações. O Algoritmo 6 apresenta o pseudocódigo do procedimento de Simulação com Matrizes e a Figura 11 ilustra o fluxograma do método.

Figura 11 – Fluxograma do método Simulação com Matrizes



Fonte: Elaborado pelo autor (2025)

Algoritmo 6: SimulaçãoComMatrizes(n, m, κ, MTPM)

```
sol_inicial ← solução de n tarefas
quantidade_matrizes ← 10.000
lista_matrizes ← []
lista_de_tarefas ← 1 to n
for i ← 1 to quantidade_matrizes do
    matriz_estocástica ← SimulandoTemposAleatórios(n, m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
L ← ListaAleatória(quantidade_matrizes, n, repetir = False)
C_máx,médio ← 0
lista_makespans ← []
for i ← 1 to n_iterações do
    C_máx,it ← CalculandoMakespanIteração(sol_inicial, n, m, L, lista_matrizes)
    lista_makespans.append(C_máx,it)
    L ← [a_n, a_1, a_2, a_3, ..., a_n−1]
end for
C_máx,médio ← sum(lista_makespans)/len(lista_makespans)
return C_máx,médio
```

5.3 META-HEURÍSTICAS COMBINADAS COM O MODELO DE REGRESSÃO LINEAR MÚLTIPLA

Nesta dissertação, a Regressão Linear Múltipla foi utilizada como modelo de aprendizagem supervisionada, combinada com as meta-heurísticas: *multi-start* e GRASP. Nessa perspectiva, constituiu-se dois modelos híbridos distintos para previsão do *makespan* estocástico, que atua como um guia para busca de novas soluções. Tal abordagem é uma extensão do trabalho desenvolvido em Souza *et al.* (2024). Destacando-se ainda, a implementação de um mecanismo de exploração de soluções inspirado no trabalho desenvolvido por Jaradat *et al.* (2016) que coleta as melhores soluções encontradas e as armazenam em uma estrutura denominada *pool*, representado por uma lista com soluções melhoradas durante o procedimento de busca local. Posteriormente, as soluções armazenadas no *pool* são submetidas a SMC com método descrito na seção (5.2) para estimar o *makespan* esperado.

A Equação (5.4) é utilizada para representar o modelo de Regressão Linear Múltipla, utilizado para estimar o *makespan* esperado.

$$\hat{y} = \beta_0 + \sum_{j=1}^{nm} \beta_j x_j \tag{5.4}$$

$\hat{y}$: *Makespan* Estocástico estimado pelo método de RLM

β_0: Termo Independente da Equação de RLM

β_j: Coeficiente Correspondente ao P_{ij}

x_j: Valor do P_{ij}

n: Número de tarefas

m: Número de máquinas

Nas abordagens MSNEHT-RLM-POOL e GRASP-RLM-POOL, foram realizados os seguintes procedimentos experimentais.

5.3.1 Geração dos Dados para Treinamento

O primeiro passo de todo processo, consistiu geração dos dados estocásticos com a finalidade de compor um *data frame* para realização do treinamento do modelo de RLM. Os dados supracitados foram gerados por meio da SMC, considerando uma distribuição Log-normal. Neste aspecto, utilizou-se a Equação (5.1) para o calcular a variância, enquanto a média e o desvio-padrão dos $\mathbb{E}[P_{ij}]$ decorrentes das instâncias, foram calculados com base nas Equações (5.2) e (5.3), respectivamente. O coeficiente para o cálculo da variância κ foi fixado em $\kappa = 0,1$. Deste modo, sucederam-se a geração dos P_{ij}, por meio de uma sequência de n tarefas, gerada aleatoriamente e por conseguinte, calculou-se o *makespan* da sequência, através da Equação (4.2), e o *makespan* médio das n_iterações, calculado pela Equação (4.3).

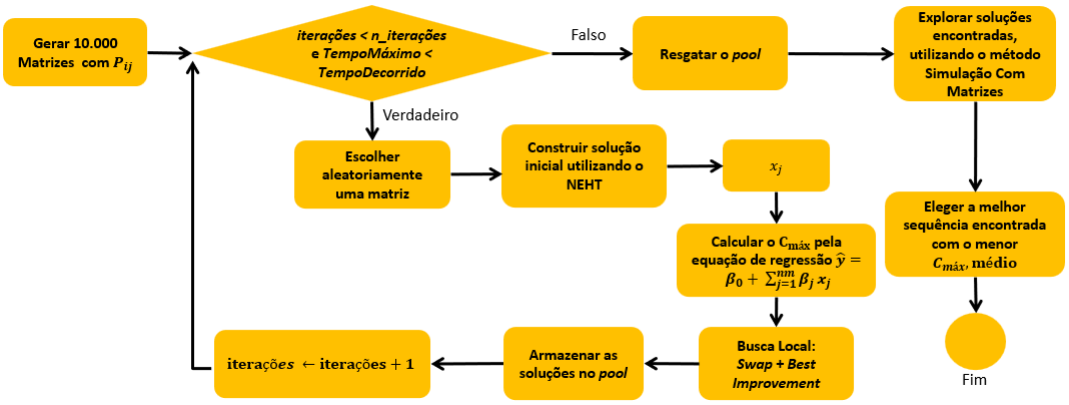
A massa de dados resultante foi consolidada em um *data frame*, no qual cada registro corresponde as seguintes informações: sequência aleatória, P_{ij} e o *makespan* médio, repetido para cada registro que leva em consideração a mesma sequência. Ou seja, a sequência foi repetida por 100 vezes (n_iterações) com os P_{ij} gerados pela SMC e o mesmo procedimento executado por 100 vezes, totalizando 10.000 registros.

5.3.2 Treinamento do Modelo de Regressão Linear Múltipla

Esta fase consistiu na aplicação do método de RLM, com o intuito de prever o *makespan* estocástico esperado, com base na Equação (5.4). A partir do *data frame* construído, excluindo-se a coluna relacionada às sequências aleatórias, separou-se as variavéis independentes P_{ij}, e a variável dependente, ou seja, o *makespan* médio. Em seguida, realizou-se a divisão do conjunto de dados, atribuindo 70% para treinamento e 30% para teste. A operação foi efetivada, utilizando recursos da biblioteca *scikit-learn*, escrita em linguagem *Python*, e o Algoritmo 7 apresenta o pseudocódigo da aplicação do modelo de RLM.

Visando avaliar o ajuste do modelo de RLM ao conjunto de dados foram calculadas as métricas: R^2, REMQ, e o EMA. Estendendo a análise a uma avaliação mais precisa, foi realizado o procedimento de validação cruzada para 10-*folds*, utilizando recursos da biblioteca *scikit-learn* e

Figura 12 – Método MSNEHT-RLM-*Pool*



Fonte: Elaborado pelo autor (2025)

Algoritmo 7: RLM(*DataFrame*)

1: variável_dependente ← Selecionar a coluna $C_{\text{máx,médio}}$ do *Data Frame*
2: variáveis_independentes ← Selecionar as demais colunas do *Data Frame*, excluindo a sequência
3: Dividir os registros do *Data Frame* em conjuntos de treinamento e teste
4: Aplicar o modelo de Regressão Linear Múltipla
5: Calcular R^2, $REMQ$, e EMA
6: Realizar validação cruzada e obter $R^2(VC)$
7: Extrair os coeficientes ← $[\beta_1, \beta_2, \beta_3, \ldots, \beta_{nm}]$ e
8: Extrair o termo independente β_0
9: **return** R^2, $R^2(VC)$, $REMQ$, EMA, coeficientes, β_0

calculando o valor médio do coeficiente de determinação para validação cruzada ($R^2(VC)$). Finalmente, foi realizada a extração dos coeficientes β_1, β_2, β_3, ..., β_{nm} e o termo independente β_0 para montagem da equação de regressão (5.4), objetivando a predição do *makespan* esperado pelo método.

5.3.3 Construção dos Modelos Híbridos com a Regressão Linear Múltipla

O desenvolvimento dos métodos híbridos com a RLM divide-se em dois blocos, que se diferenciam, basicamente, pela solução construtiva. O primeiro método usa a meta-heurística *multi-start* com uma solução construtiva elaborada por meio da heurística NEHT, calculada com auxílio da biblioteca *permutation-flowshop*, cujo desenvolvimento está atrelado a esta dissertação como produto tecnológico, registrado no Instituto Nacional de Propriedade Industrial (INPI), sob o Nº BR512024004524-4. E o segundo emprega a meta-heurística GRASP com solução construtiva baseada na LCR. A seguir, serão retratadas as etapas comuns aos dois métodos.

A geração dos P_{ij} fornecida aos modelos seguiu a ideia da abordagem de simulação proposta neste trabalho, discutida na Seção 5.2. Construindo-se uma adaptação ao método genérico supracitado, no qual realizou-se a geração de 10.000 matrizes com os tempos de n tarefas e a geração dos P_{ij} de cada tarefa i em cada máquina j. Com a realização da seleção aleatória das 10.000 matrizes

com tempos de processamento simulados armazenadas em uma lista *L*.

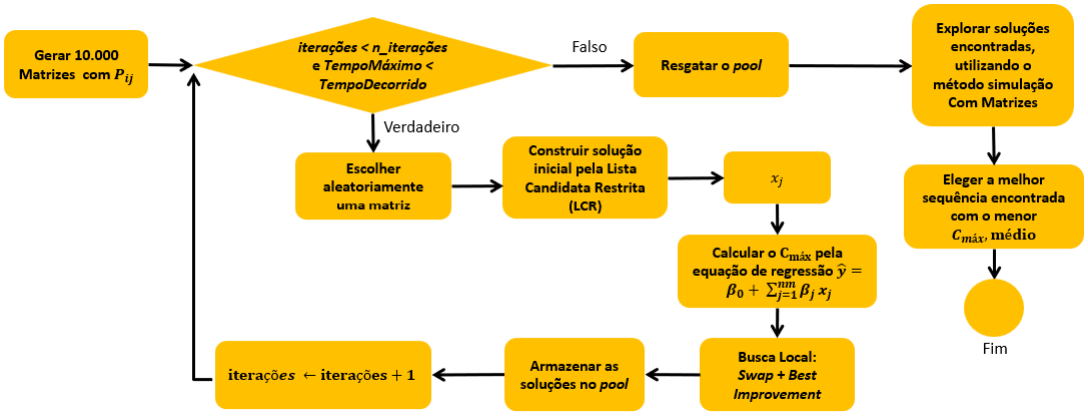
Os tempos simulados nas matrizes serão utilizados como entrada para a solução construtiva e representarão uma simulação para compôr as variáveis x_j, necessárias ao modelo para estimar o *makespan* esperado $\hat{y}$, calculado pela Equação de Regressão (5.4). O mesmo procedimento é aplicado para o modelo que utiliza a meta-heurística GRASP.

A segunda parte dos métodos, engloba a execução do procedimento de busca local utilizando o operador de vizinhança *swap* com a estratégia *best improvement*, estimando o *makespan* esperado por meio da regressão a cada permutação das tarefas na sequência inicial, onde o valor corrente foi substituído pelo menor valor encontrado e adiciona-se a sequência melhorada a uma lista de melhores soluções (*pool*).

Após o término da busca local é realizada o uso do procedimento que envolve 500 SMC para cada uma das soluções armazenadas no *pool*, a fim de se obter uma melhor estimativa para o *mekespan* esperado. Tal procedimento, leva em consideração a metodologia abordada na Seção 6.

O Algoritmo 8 apresenta o pseudocódigo MSNEHT-RLM-POOL, enquanto o Algoritmo 9 mostra o pseudocódigo GRASP-RLM-POOL, ambos desenvolvidos para estimar o *makespan*. O Algoritmo 8 representa o pseudocódigo associado ao método MSNEHT-RLM-POOL e o Algoritmo 9 desenvolve o pseudocódigo relacionado ao modelo GRASP-RLM-POOL. As Figuras 12 e 13 representa o fluxo associado, respectivamente aos métodos supracitados. Ressaltando-se ainda, que o Algoritmo 10 representa o pseudocódigo a solução construtiva GRASP, baseada na LCR e o Algoritmo 11 representa o pseudocódigo do procedimento de busca local implementado em ambos.

Figura 13 – Método GRASP-RLM-*Pool*



Fonte: Elaborado pelo autor (2025)

Algoritmo 8: MSNEHT-RLM (κ, coeficientes, β_0, MTPM, n, m)

máx_iteração ← 10
Iter ← 0
melhores_soluções_multistart ← []
tempo_máximo ← 3600
quantidade_matrizes ← 10.000
lista_matrizes ← []
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter < máx_iteração **do**
    matriz_aleatória ← SelecionandoÍndicesAleatório(1, quantidade_matrizes)
    P_{ij} ← lista_matrizes[matriz_aleatória]
    $sol\_inicial$ ← NEHT(n,m, P_{ij})
    x_j ← Calcular_x_j($sol\_inicial$, P_{ij}, n,m)
    makespan_regressão ← CalcularMakespanRegressão(x_j, $coeficientes$, β_0)
    melhores_soluções_busca_local ← BuscaLocalML($sol\_inicial$, n, m, makespan_regressão, P_{ij})
    melhores_soluções_multistart.append(melhores_soluções_busca_local)
    $Iter \leftarrow Iter + 1$
end while
melhor_sequência_$pool$ ← []
$C_{\text{máx, médio}} \leftarrow 0$
for solução ∈ melhores_soluções_multistart **do**
    lista_makespans ← []
    número_simulacões ← 500
    L ← ÍndicesAleatórios(quantidade_matrizes, n, repetir=False)
    for _ ← 1 **to** número_simulações **do**
        $C_{\text{máx,it}}$ ← CalcularMakespan(solução, n, m, L, lista_de_matrizes)
        lista_makespans.append($C_{\text{máx,it}}$)
        $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
    end for
    melhor_$C_{\text{máx}}$ ← ∞
    $C_{\text{máx, médio}}$ ← `sum(lista_makespans) / len(lista_makespans)`
    if $C_{\text{máx, médio}}$ < $melhor\_C_{\text{máx}}$ **then**
        $melhor\_C_{\text{máx}} \leftarrow C_{\text{máx, médio}}$
        melhor_sequência_$pool$ ← solução
    end if
end for
return melhor_sequência_$pool$, $melhor\_C_{\text{máx}}$

5.4 MODELOS BASEADOS NA APRENDIZAGEM POR REFORÇO

Os algoritmos que serão abordados nesta seção, envolve a segunda abordagem proposta, desenvolvida através das modificações realizadas nos modelos relatados e proposto por He *et al.* (2022). Essas modificações foram executadas no que tange a inserção de estocasticidade e a forma de seleção da melhor solução dos modelos. Considerando a natureza combinatória do problema estudado e tendo em vista que empiricamente a melhor sequência pode ser encontrada ainda durante a fase de execução

Algoritmo 9: GRASP-RLM (κ, coeficientes, β_0, MTPM, n, m)

$\lambda \leftarrow 0.5$
máx_iteração $\leftarrow 10$
Iter $\leftarrow 0$
melhores_soluções_multistart $\leftarrow$ []
tempo_máximo $\leftarrow 3600$
quantidade_matrizes $\leftarrow 10.000$
lista_matrizes $\leftarrow$ []
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica $\leftarrow$ SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter < máx_iteração **do**
    matriz_aleatória $\leftarrow$ SelecionandoÍndicesAleatórios(1, quantidade_matrizes)
    $P_{ij} \leftarrow$ lista_matrizes[matriz_aleatória]
    $sol\_inicial \leftarrow$ ConstrutivaGrasp(λ, n, m, P_{ij})
    $x_j \leftarrow$ Calcular_x_j($sol\_inicial$, P_{ij}, n,m)
    makespan_regressão $\leftarrow$ CalcularMakespanRegressão(x_j, $coeficientes$, β_0)
    melhores_soluções_busca_local $\leftarrow$ BuscaLocalML($sol\_inicial$, n, m, makespan_regressão, P_{ij})
    melhores_soluções_multistart.append(melhores_soluções_busca_local)
    $Iter \leftarrow Iter + 1$
end while
melhor_sequência_$pool$ $\leftarrow$ []
$C_{\text{máx, médio}} \leftarrow 0$
for solução $\in$ melhores_soluções_multistart **do**
    lista_makespans $\leftarrow$ []
    número_simulacões $\leftarrow 500$
    $L \leftarrow$ ÍndicesAleatórios(quantidade_matrizes, n, repetir=False)
    for _ $\leftarrow 1$ **to** número_simulações **do**
        $C_{\text{máx,it}} \leftarrow$ CalcularMakespan(solução, n, m, L, lista_de_matrizes)
        lista_makespans.append($C_{\text{máx,it}}$)
        $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
    end for
    melhor_$C_{\text{máx}} \leftarrow \infty$
    $C_{\text{máx, médio}} \leftarrow$ `sum(lista_makespans) / len(lista_makespans)`
    if $C_{\text{máx, médio}} <$ $melhor\_C_{\text{máx}}$ **then**
        $melhor\_C_{\text{máx}} \leftarrow C_{\text{máx, médio}}$
        melhor_sequência_$pool$ $\leftarrow$ solução
    end if
end for
return melhor_sequência_$pool$, $melhor\_C_{\text{máx}}$

dos episódios, nas abordagens aqui propostas, esse ponto foi adotado como premissa. A seguir, serão apresentados os algoritmos propostos nesta dissertação *Q-Learning NEH - Simulation* (QLNEH-S) e o *Improvement Q-Learning - Simulation* (IQL-S).

Os métodos QLNEH-S e IQL-S inspirados, respectivamente nos algoritmos QLNEH e IQL (He *et al.*, 2022), estes originalmente desenvolvidos abordando para análise de tempos médios. Os métodos aqui proposto os modificam em alguns pontos com destaque para inserção da SMC

Algoritmo 10: ConstrutivaGRASP(λ, n, m, P_{ij})

```
sol_construtiva ← []
LCR ← []
C ← [1 ... n]
matriz_makespans ← matriz m x n de zeros
makespan_incremental ← lista de zeros com dimensão n
while C do
    lista_makespans_tarefas = []
    for i ∈ C do
        Calcular o C_máx,incremental da tarefa i com base nos P_ij
        lista_makespans_tarefas.append(C_máx,incremental[i])
    end for
    g_min ← min(lista_makespans_tarefas)
    g_max ← max(lista_makespans_tarefas)
    for i ∈ C do
        if makespan_incremental[i] ≤ (g_min + λ · (g_max − g_min)) then
            LCR.append(i)
        end if
    end for
    if LCR then
        Selecionar aleatoriamente uma tarefa de LCR
        Adicionar a tarefa selecionada à solução sol_construtiva
        Remover a tarefa selecionada de C
        LCR ← []
    end if
end while
```

que também seguindo a abordagem discutida na Seção (5.2) para estimar um *makespan* esperado. Outro ponto, relevante refere-se a forma de seleção do valor estimado e a sequência associada, estes durante a execução dos episódios dos algoritmos, quando as alterações da matriz Q e o mecanismo de explotação-exploração podem levar a uma solução melhor que a solução encontrada ao final da execução de todos os episódios.

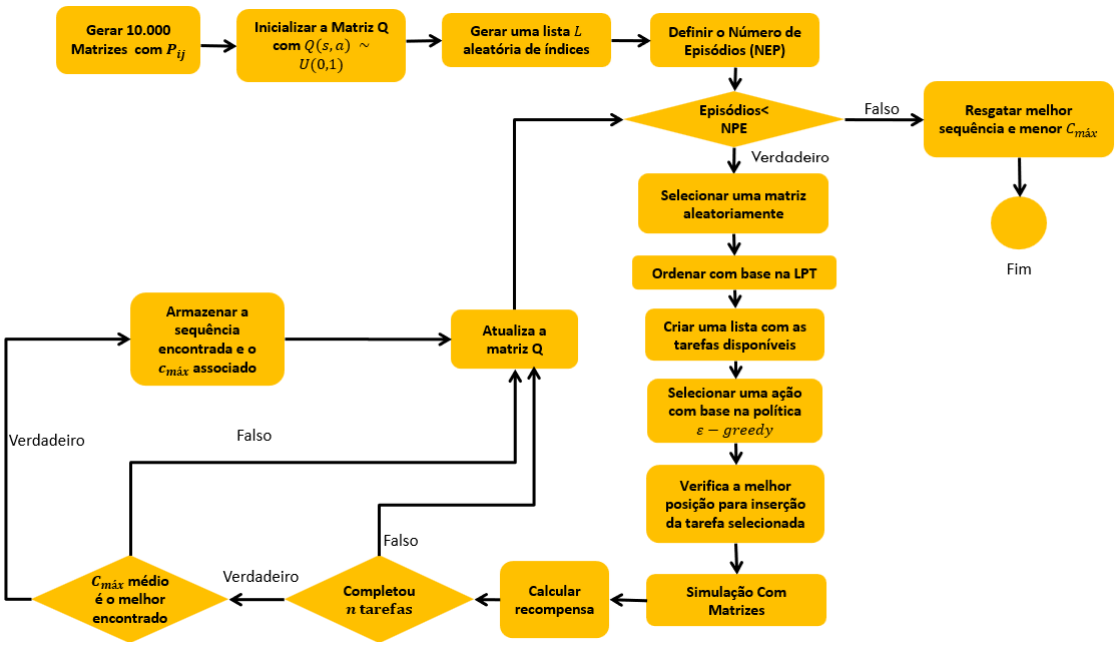
O Algoritmo 12 , apresenta o pseudocódigo do IQL-S e o Algoritmo 13 , mostra o pseudocódigo do QLNEH-S. Os parâmetros necessários para execução dos métodos são: os tempos de processamento médio (MTPM), um coeficiente para o cálculo da variância κ, utilizado para o cálculo da simulação do tempo estocástico baseado na simulação de Monte Carlo, número de tarefas n, número de máquinas m, taxa de aprendizagem θ, a taxa de desconto γ, um valor probabilístico seguindo a estratégia $\varepsilon - greedy$ que irá determinar o controle entre exploração e explotação, ou seja ao gerar um valor aleatório entre 0 e 1, seguindo uma distribuição uniforme, se esse for maior $\varepsilon >$ valor definido o algoritmo irá realizar a exploração, escolhendo uma tarefa aleatória, caso contrário a escolha da próxima tarefa será com a explotação, ou seja, baseada nos valores da matriz Q.

A Tabela 2 representa a ideia geral da matriz Q de recompensas. Na qual é atribuído um

Algoritmo 11: BuscaLocalML(sol_inicial, n, m, makespan_regressão, P_{ij})

```
sol ← sol_inicial
melhor_makespan_bl ← makespan_regressão
melhoria ← True
melhores_soluções_bl ← []
while melhoria == True do
    melhoria ← False
    for i ← 1 to n do
        for j ← i + 1 to n do
            if tempo_decorrido_busca ≥ tempo_restante then
                break
            end if
            sequência_swap ← Swap(sol_inicial)
            x_j ← Calcular_x_j(sequência_swap, P_ij, n,m)
            makespan_esperado_swap ← CalcularMakespanRegressão(x_j, coeficientes, β_0)
            if makespan_esperado_swap < melhor_makespan_bl then
                melhor_makespan_bl ← makespan_esperado_swap
                sol ← sequência_swap
                melhores_soluções_bl.append(sol)
                melhoria ← True
            end if
        end for
    end for
end while
return melhores_soluções_bl
```

Figura 14 – Método QLNEH-S

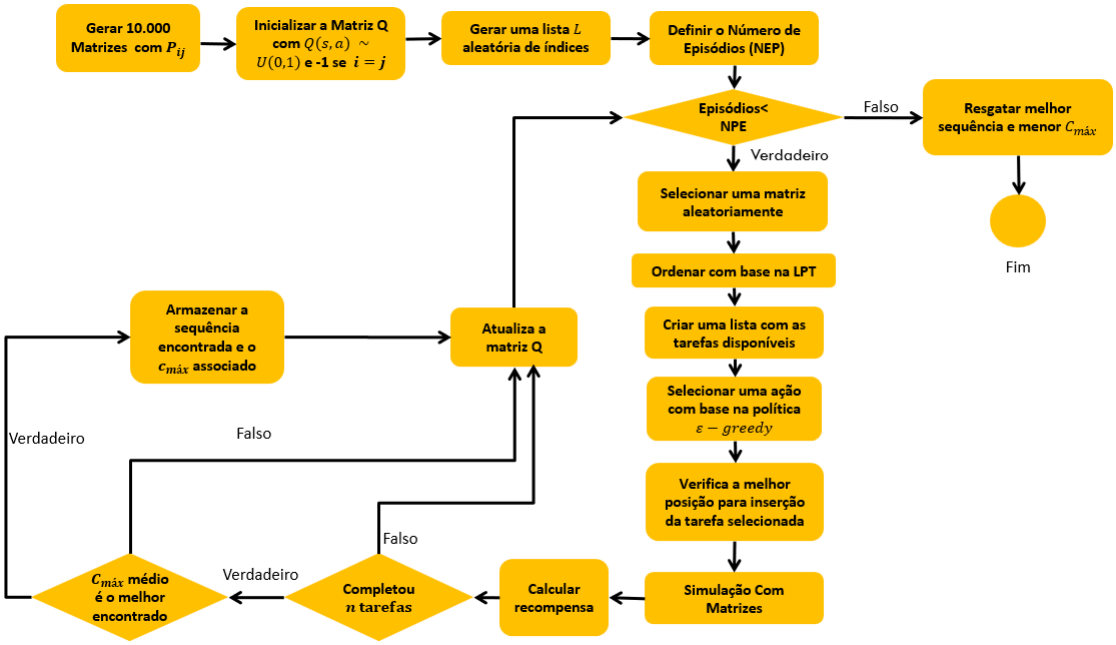


Fonte: Elaborado pelo autor (2025)

valor de -1 para cada tarefa T_n. Ou seja, a recompensa de uma tarefa ir para ela mesma fica negativa.

A lógica do QL implica na necessidade de elaboração de uma função que estabeleça uma relação que vise maximizar a recompensa. Neste trabalho, a função de recompensa está baseada no

Figura 15 – Método IQL-S



Fonte: Elaborado pelo autor (2025)

Tabela 1 – Representação da Matriz Q de recompensas

Estado/Ação	a_1	a_2	a_3	...	a_n
s_1	Q(s_1, a_1)	Q(s_1, a_2)	Q(s_1, a_3)	...	Q(s_1, a_n)
s_2	Q(s_2, a_1)	Q(s_2, a_2)	Q(s_2, a_3)	...	Q(s_2, a_n)
s_3	Q(s_3, a_1)	Q(s_3, a_2)	Q(s_3, a_3)	...	Q(s_3, a_n)
...	...	...	...	...	...
s_n	Q(s_n, a_1)	Q(s_n, a_2)	Q(s_3, a_n)	...	Q(s_n, a_n)

Fonte: Adaptado de He *et al.* (2022)

Tabela 2 – Representação da Matriz Q do IQL

Estado/Ação	T_1	T_2	T_3	...	T_n
T_1	-1	Q(T_1, T_2)	Q(T_1, T_3)	...	Q(T_1, T_n)
T_2	Q(T_2, T_1)	-1	Q(T_2, T_3)	...	Q(T_2, T_n)
T_3	Q(T_3, T_1)	Q(T_3, T_2)	-1	...	Q(T_3, T_n)
...	...	...	...	-1	...
T_n	Q(T_n, T_1)	Q(T_n, T_2)	Q(T_3, T_n)	...	-1

Fonte: Adaptado de He *et al.* (2022)

makespan da sequência parcial, tal qual função utilizada por He *et al.* (2022). A Equação 5.5 representa a função para o cálculo da recompensa a cada seleção de tarefa, esta função é indicada como inverso do $C_{\text{máx}}(\text{sequência\_parcial})$ parcial da sequência.

$$r = \frac{1}{C_{\text{máx}}(\text{sequência\_parcial})} \tag{5.5}$$

Foi considerado que valor aleatório $\sim$ U(0,1), logo para escolha da ação tomada (a_t)

$$a_t = \begin{cases} \text{aleatória, se valor aleatório} > \varepsilon, \\ \arg\max(Q), \text{ c.c.} \end{cases}$$

5.5 META-HEURÍSTICA *MULTI-START* COMBINADA COM MODELOS DE APRENDIZAGEM POR REFORÇO

Nesta seção serão abordados os algoritmos híbridos que relacionam a aprendizagem por reforço e a meta-heurística *multi-start*, nomeados de *Multi-start Q-Learning NEH - Stochastic* (MSQLNEH-S) e *Multi-start Improvement Q-Learning - Stochastic* (MSIQLS-S). A estratégia de ambos os algoritmos consiste em utilizar o QLNEH-S e o IQL-S como método de geração da solução construtiva para execução da meta-heurística *multi-start* com intuito de encontrar soluções estocásticas para o problema do *flow shop* permutacional, pressupondo que as soluções construídas pelos métodos supracitados, sejam soluções melhores que uma solução aleatória e com isso durante o procedimento de busca local, haja uma melhora da solução inicial e consequentemente uma solução final melhor para o problema de sequenciamento.

As duas meta-heurísticas supramencionadas tem os pseudocódigo apresentados no Algoritmo 14 e no Algoritmo 15. A elaboração dessas meta-heurísticas híbridas envolveu uma modificação nos parâmetros de treinamento dos algoritmos QLNEH-S e IQL-S, mais especificamente no que corresponde ao treinamento dos algoritmos. A mudança se deu em relação a quantidade de episódios de treinamento, isso foi realizado devido ao fato do tempo necessário para treinar grandes instâncias ser consideravelmente alto, o que impactaria diretamente no tempo de execução da meta-heurística.

5.6 SIMHEURÍSTICAS

Neste trabalho, ainda foram desenvolvidas duas simheurísticas equivalentes, que, com o intuito de aumentar a eficiência dos métodos, fazem uso do modelo de Simulação com Matrizes, em vez de usar a simulação clássica, denominadas MSNEHT-SIMH e GRASP-SIMH. Cujo a ideia, centra-se na realização de uma simulação curta (200 iterações) para calcular o $C_{\text{máx}}$ esperado da solução construtiva e durante o processo da busca local. O Algoritmo 16 representa o pseudocódigo do método MSNEHT-SIMH, enquanto o Algoritmo 17 apresenta o pseudocódigo do método GRASP-SIMH. Ambos possuem, o mesmo procedimento de busca local, mostrado no Algoritmo 18. As Figuras 16 e 17, apresentam os respectivos fluxogramas associados aos métodos MSNEHT-SIMH e GRASP-SIMH.

Figura 16 – Método MSNEHT-SIMH

Fonte: Elaborado pelo autor (2025)

Figura 17 – Método GRASP-SIMH

Fonte: Elaborado pelo autor (2025)

5.7 META-HEURÍSTICAS CONSIDERANDO OS TEMPOS MÉDIOS

Prosseguindo no desenvolvimento de meios para comparação dos resultados dos métodos propostos, foram construídas duas meta-heurísticas considerando apenas os tempos médios, ou seja, sem que ocorra simulações estocástica durante o processo. Tais métodos foram nomeados de NEHTBLMED e GRASPMED. As Figuras 18 e 19, respectivamente, representam o fluxograma ilustrativos desses métodos.

O NEHTBLMED é o método mais simples desenvolvido neste trabalho e consiste na execução da heurística NEHT para a MTPM de cada instância e em seguida é realizada uma busca local na sequência encontrada com o operador de vizinhança *swap* e a estratégia *best improvement*. Sob a mesma ótica, foi desenvolvido o método GRASPMED, com a construtiva baseada na LCR, usando o $\lambda = 0.5$ e como critérios de paradas 10 iterações ou 3600 segundos. O Algoritmo 19 apresenta o pseudocódigo do método NEHTBLMED e o Algoritmo 20 mostra o pseudocódigo do modelo GRASPMED. Por fim, o Algoritmo 21 demonstra o procedimento de busca local aplicado a ambos métodos.

Figura 18 – NEHTBLMED

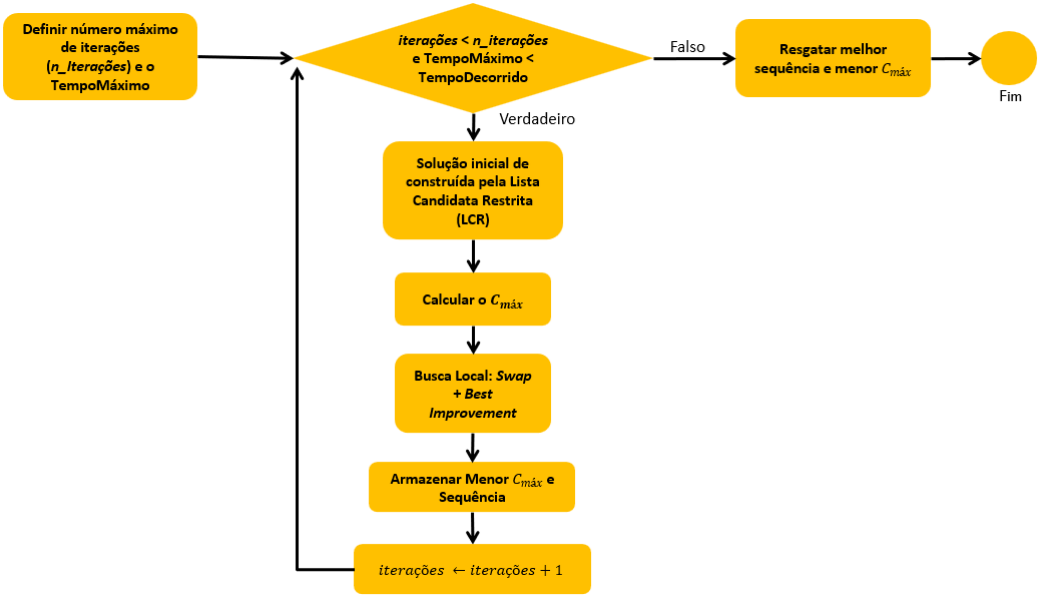
Fonte: Elaborado pelo autor (2025)

5.8 CRITÉRIOS DOS PROCEDIMENTOS EXPERIMENTAIS

Os critérios de parada adotados para as meta-heurísticas híbrida desenvolvidas neste trabalho foram 10 iterações ou 3600 segundos (1 hora), tempo máximo escolhido para execução de cada procedimento. O método NEHTBLMED é executado apenas uma vez, haja vista que os tempos médios não demandam simulação e caso estivesse inserido na meta-heurística a cada *start* iria começar sempre da mesma solução.

Os algoritmos que envolvem a aprendizagem por reforço QLNEH-S e IQL-S , foram

Figura 19 – GRASPMED



Fonte: Elaborado pelo autor (2025)

parametrizados igualmente e os valores atribuídos aos parâmetros estão apresentados na Tabela 3. Onde, θ é a taxa de aprendizagem, γ é a taxa de desconto, ε refere-se a probabilidade de realizar exploração ou explotação e $N_{\text{episódios}}$ representa o número de execuções dos procedimentos associados.

Como guia de precisão, ao final da obtenção de cada solução associada a cada instância e método, foi aplicado uma simulação para melhor sequência encontrada em uma simulação longa de 10.000 iterações, utilizando como método a Simulação Clássica.

Para os algoritmos híbridos que unem a aprendizagem por reforço e a meta-heurística *multi-start*. Os parâmetros foram os mesmo apresentados na Tabela 3, com exceção do $N_{\text{episódios}}$ que foram definidos apenas 10, haja vista que para instâncias maiores 100 episódios demanda um tempo razoavelmente maior e considerando que na metodologia adotada, os métodos IQL-S e QLNEH-S foram utilizados apenas como solução construtiva, partindo da premissa que a busca local poderia melhorar consideravelmente as soluções iniciais encontradas pelos métodos.

Algoritmo 12: IQL-S (κ, $N_{\text{episódios}}$, θ, γ, ε, MTPM, n, m)

Matriz Q: $n \times n$
quantidade_matrizes ← 10.000
lista_matrizes ← []
melhor_sequência_absoluta ← []
menor_makespan_absoluto ← ∞
número_simulações ← 10
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    if $i == j$ **then**
      Q[i][j] ← -1
    else
      Q[i][j] ∼ Uniforme(0,1)
    end if
  end for
end for
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
  matriz_estocástica ← SimulandoTemposAleatórios(n, m, κ, MTPM)
  lista_matrizes.append(matriz_estocástica)
end for
for episódio = 1 **to** $N_{\text{episódios}}$ **do**
  $L \leftarrow$ ÍndicesAleatórios(quantidade_matrizes, n, repetir=False)
  matriz_aleatória ← SelecionandoÍndicesAleatório(1, quantidade_matrizes)
  $P_{ij} \leftarrow$ lista_matrizes[matriz_aleatória]
  Calcular a soma dos P_{ij}
  Ordenar as tarefas com base na *Longest Processing Time* (LPT)
  Alocar as duas primeiras tarefas na melhor_sequência
  LAP ← tarefas que não estão na melhor_sequência
  if valor aleatório ∼ Uniforme(0,1) > ε **then**
    escolhe uma ação aleatória em LAP
  else
    Escolhe uma ação com base no maior valor (estado, ação) da matriz Q
  end if
  Insere a ação escolhida na posição que apresentou menor $C_{\text{máx}}$, considerando P_{ij} e as matrizes escolhidas aleatoriamente em lista_matrizes
  lista_makespans ← []
  $C_{\text{máx, médio}} \leftarrow 0$
  for _ ← 1 **to** número_simulações **do**
    $C_{\text{máx, it}} \leftarrow$ CalcularMakespan(melhor_sequência, n, m, L, lista_matrizes)
    lista_makespans.append($C_{\text{máx, it}}$)
    $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
  end for
  $C_{\text{máx, médio}} \leftarrow$ `sum(lista_makespans) / len(lista_makespans)`
  recompensa ← 1 / $C_{\text{máx, médio}}$
  if len(melhor_sequência) == n **then**
    if $C_{\text{máx, médio}}$ < menor_makespan_absoluto **then**
      menor_makespan_absoluto ← $C_{\text{máx, médio}}$
      melhor_sequência_absoluta ← melhor_sequência
    end if
  end if
  Atualiza a matriz Q(estado, ação)
end for
return melhor_sequência_absoluta, menor_makespan_absoluto

Algoritmo 13: QLNEH-S (κ, $N_{\text{episódios}}$, θ, γ, ε, MTPM, n, m)

Matriz Q: $n \times n$
quantidade_matrizes ← 10.000
lista_matrizes ← []
melhor_sequência_absoluta ← []
menor_makespan_absoluto ← ∞
número_simulações ← 10
for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
        Q[i][j] ∼ Uniforme(0,1)
    end for
end for
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n, m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
for episódio = 1 **to** $N_{\text{episódios}}$ **do**
    L ← índices_aleatórios(quantidade_matrizes, n, repetir=False)
    matriz_aleatória ← SelecionandoÍndicesAleatório(1, quantidade_matrizes)
    P_{ij} ← lista_matrizes[matriz_aleatória]
    Calcular a soma dos P_{ij}
    Ordenar as tarefas com base na *Longest Processing Time* (LPT)
    Alocar as duas primeiras tarefas na melhor_sequência
    LAP ← tarefas que não estão na melhor_sequência
    if valor aleatório ∼ Uniforme(0,1) > ε **then**
        Escolhe uma ação aleatória em LAP
    else
        Escolhe uma ação com base no maior valor (estado, ação) da matriz Q
    end if
    Insere a ação escolhida na posição que apresentou menor $C_{\text{máx}}$, considerando P_{ij} e as matrizes escolhidas aleatoriamente em lista_matrizes
    lista_makespans ← []
    $C_{\text{máx, médio}} \leftarrow 0$
    for _ ← 1 **to** número_simulações **do**
        $C_{\text{máx, it}}$ ← CalcularMakespan(melhor_sequência, n, m, L, lista_matrizes)
        lista_makespans.append($C_{\text{máx, it}}$)
        $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
    end for
    $C_{\text{máx, médio}}$ ← `sum(lista_makespans) / len(lista_makespans)`
    recompensa ← 1 / $C_{\text{máx, médio}}$
    if len(melhor_sequência) == n **then**
        if $C_{\text{máx, médio}}$ < menor_makespan_absoluto **then**
            menor_makespan_absoluto ← $C_{\text{máx, médio}}$
            melhor_sequência_absoluta ← melhor_sequência
        end if
    end if
    Atualiza a matriz Q(estado, ação)
end for
return melhor_sequência_absoluta, menor_makespan_absoluto

Algoritmo 14: MSIQL-S(κ, $N_{\text{episódios}}$, θ, γ, ε, MTPM, n, m)

máx_iteração ← 10
menor_makespan ← ∞
Iter ← 0
melhor_sequência ← []
tempo_máximo ← 3600
quantidade_matrizes ← 10.000
lista_matrizes ← []
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter <máx_iteração **do**
    sol_inicial, makespan_inicial ← IQL-S(κ, $N_{\text{episódios}}$, θ, γ, ε, TPM_{ij}, n, m, quantidade_matrizes, lista_de_matrizes)
    sequência_bl, makespan_bl ← BuscaLocalSimh(sol_inicial, n, $C_{\text{máx, médio}}$, lista_de_matrizes, quantidade_matrizes)
    if makespan_bl < menor_makespan **then**
        menor_makespan ← makespan_bl
        melhor_sequência ← sequência_bl
    end if
    $Iter \leftarrow Iter + 1$
end while
return melhor_sequência, menor_makespan

Algoritmo 15: MSQLNEH-S(κ, $N_{\text{episódios}}$, θ, γ, ε, MTPM, n, m)

máx_iteração ← 10
menor_makespan ← ∞
Iter ← 0
melhor_sequência ← []
tempo_máximo ← 3600
quantidade_matrizes ← 10.000
lista_matrizes ← []
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter <máx_iteração **do**
    sol_inicial, makespan_inicial ← QLNEH-S(κ, $N_{\text{episódios}}$, θ, γ, ε, MTPM, n, m, quantidade_matrizes, lista_de_matrizes)
    sequência_bl, makespan_bl ← BuscaLocalSimh(sol_inicial, n, $C_{\text{máx, médio}}$, lista_de_matrizes, quantidade_matrizes)
    if makespan_bl < menor_makespan **then**
        menor_makespan ← makespan_bl
        melhor_sequência ← sequência_bl
    end if
    $Iter \leftarrow Iter + 1$
end while
return melhor_sequência, menor_makespan

Algoritmo 16: MSNEHT-SIMH($\kappa, MTPM, n, m$)

máx_iteração ← 10
menor_$C_{máx}$ ← ∞
Iter ← 0
melhor_sequência ← []
tempo_máximo ← 3600
quantidade_matrizes ← 10.000
lista_matrizes ← []
for $i \leftarrow 1$ **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter < máx_iteração **do**
    matriz_aleatória ← SelecionandoÍndicesAleatórios(1, quantidade_matrizes)
    P_{ij} ← lista_matrizes[matriz_aleatória]
    sol_inicial ← NEHT(n,m, P_{ij})
    lista_makespans ← []
    $C_{máx,\ médio}$ ← 0
    número_simulacões ← 200
    for _ ← 1 **to** número_simulações **do**
        $C_{máx,it}$ ← CalcularMakespan(sol_inicial, n, m, L, lista_de_matrizes)
        lista_makespans.append($C_{máx,it}$)
        $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
    end for
    $C_{máx,\ médio}$ ← `sum(lista_makespans) / len(lista_makespans)`
    sequência_bl, makespan_bl ← BuscaLocalSimh(sol_inicial, n, $C_{máx,\ médio}$, lista_de_matrizes, quantidade_matrizes)
    if $C_{máx,\ médio}$ < *menor_*$C_{máx}$ **then**
        *menor_*$C_{máx}$ ← makespan_bl
        melhor_sequência ← sequência_bl
    end if
    $Iter \leftarrow Iter + 1$
end while
return melhor_sequência, *menor_*$C_{máx}$

Algoritmo 17: GRASP-SIMH(κ, λ, MTPM, n, m)

máx_iteração ← 10
menor_$C_{máx}$ ← ∞
Iter ← 0
melhor_sequência ← []
tempo_máximo ← 3600
quantidade_matrizes ← 10.000
lista_matrizes ← []
for i ← 1 **to** quantidade_matrizes **do**
    matriz_estocástica ← SimulandoTemposAleatórios(n,m, κ, MTPM)
    lista_matrizes.append(matriz_estocástica)
end for
while tempo_decorrido < tempo_máximo e Iter < máx_iteração **do**
    matriz_aleatória ← SelecionandoÍndicesAleatórios(1, quantidade_matrizes)
    P_{ij} ← lista_matrizes[matriz_aleatória]
    sol_inicial ← ConstrutivaGRASP(λ, n,m, P_{ij})
    lista_makespans ← []
    $C_{máx,\ médio}$ ← 0
    número_simulacões ← 200
    for _ ← 1 **to** número_simulações **do**
        $C_{máx,it}$ ← CalcularMakespan(sol_inicial, n, m, L, lista_de_matrizes)
        lista_makespans.append($C_{máx,it}$)
        $L \leftarrow [a_n, a_1, a_2, a_3, \ldots, a_{n-1}]$
    end for
    $C_{máx,\ médio}$ ← `sum(lista_makespans) / len(lista_makespans)`
    sequência_bl, makespan_bl ← BuscaLocalSimh(sol_inicial, n, $C_{máx,\ médio}$, lista_de_matrizes, quantidade_matrizes)
    if $C_{máx,\ médio}$ < menor_$C_{máx}$ **then**
        menor_$C_{máx}$ ← makespan_bl
        melhor_sequência ← sequência_bl
    end if
    $Iter \leftarrow Iter + 1$
end while
return melhor_sequência, *menor*_$C_{máx}$

Algoritmo 18: BuscaLocalSimH(sol_inicial,$C_{\text{máx, inicial}}$, n, m, MTPM)

```
sol ← sol_inicial
menor_makespan_bl ← C_máx, inicial
melhor_sequência_bl ← sol_inicial
melhoria ← True
número_simulações ← 200
L ← ÍndicesAleatórios(quantidade_matrizes, n, repetir=False)
while melhoria == True do
    melhoria ← False
    for i to n do
        for j = i+1 to n do
            sol[i], sol[j] = sol[j], sol[i]
            C_máx, médio ← 0
            lista_makespans ← []
            for _ ← 1 to número_simulações do
                C_máx,it ← CalcularMakespan(sol, n, m, L, lista_de_matrizes)
                lista_makespans.append(C_máx,it)
                L ← [a_n, a_1, a_2, a_3, ..., a_n−1]
            end for
            if C_máx < menor_makespan_bl then
                menor_makespan_bl ← C_máx
                melhor_sequência ← sequencia_bl
                melhor_i ← i
                melhor_j ← j
                melhoria ← True
            end ifsol[i], sol[j] = sol[j], sol[i]
        end for
        if melhorou then
            sol[melhor_i], sol[melhor_j] = sol[melhor_j], sol[melhor_i]
        end if
    end for
end while
return sol, menor_makespan_bl
```

Algoritmo 19: NEHTBLMED(n, m, MTPM)

```
menor_makespan ← ∞
melhor_sequência ← []
sol_inicial ← NEHT(n,m, MTPM)
C_máx, inicial ← CalcularMakespan(sol_inicial, n,m, MTPM)
sequência_bl, makespan_bl ← BuscaLocalMed(sol_inicial,C_máx, inicial, n, m, MTPM)
if makespan_bl < menor_makespan then
    menor_makespan ← makespan_bl
    melhor_sequência ← sequencia_bl
end if
return melhor_sequência, menor_makespan
```

Algoritmo 20: GRASPMED(n, m, MTPM)

```
menor_makespan ← ∞
melhor_sequência ← []
sol_inicial ← ConstrutivaGraspMed(λ, n,m, MTPM)
C_máx, inicial ← CalcularMakespan(sol_inicial, n,m, MTPM)
sequência_bl, makespan_bl ← BuscaLocalMed(sol_inicial,C_máx, inicial, n, m, MTPM)
if makespan_bl < menor_makespan then
    menor_makespan ← makespan_bl
    melhor_sequência ← sequência_bl
end if
return melhor_sequência, menor_makespan
```

Algoritmo 21: BuscaLocalMed(sol_inicial,$C_{\text{máx, inicial}}$, n, m, MTPM)

```
sol ← sol_inicial
menor_makespan_bl ← C_máx, inicial
melhor_sequência_bl ← sol_inicial
melhoria ← True
while melhoria == True do
    melhoria ← False
    for i to n do
        for j = i+1 to n do
            sol[i], sol[j] = sol[j], sol[i]
            C_máx ← CalcularMakespan(sol, n,m, MTPM)
            if C_máx < menor_makespan_bl then
                menor_makespan_bl ← C_máx
                melhor_sequência ← sequencia_bl
                melhor_i ← i
                melhor_j ← j
                melhoria ← True
            end ifsol[i], sol[j] = sol[j], sol[i]
        end for
        if melhorou then
            sol[melhor_i], sol[melhor_j] = sol[melhor_j], sol[melhor_i]
        end if
    end for
end while
return sol, menor_makespan_bl
```

Tabela 3 – Parâmetros de Treinamento para os Métodos QLNEH-S e IQL-S

Parâmetro	**Valor**
θ	0,2
γ	0,85
ε	0,75
$N_{\text{episódios}}$	100

Fonte: Elaborada pelo autor (2025)

6 RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados obtidos para os métodos propostos e implementados. O capítulo está dividido da seguinte forma: a primeira parte descreve a caracterização dos dados de treinamento do modelo de regressão linear múltipla; a segunda parte apresenta as métricas obtidas pelo modelo de regressão e o tempo necessário para treinamento; a terceira parte discorre sobre a eficácia dos métodos desenvolvidos; e a quarta parte apresenta o simulador implementado.

Os experimentos computacionais foram realizados em um computador com as seguintes especificações técnicas: processador Intel Core i5-1235U, 12GB de memória RAM e sistema operacional Linux Mint 22.

O detalhamento quanto aos dados utilizados para treinamento do modelo de Regressão Linear Múltipla, encontra-se disposto no Apêndice A - CARACTERIZAÇÃO DOS DADOS DE TREINAMENTO. As Figuras 34, 35, 36 e 37 apresentam os histogramas para os *makespans* médio dos conjuntos de dados estocásticos gerados pela SMC para construção do *data frame*, considerando 10 *bins*. Ressaltando-se que a menor frequência absoluta do conjunto de dados é de 100 unidades, haja vista que os P_{ij} associados a uma mesma sequência são simulados 100 vezes (gerando um *makespan* médio para 100 entradas).

As Figuras 38, 39 e 40 mostram os gráficos *box plots* das distribuição dos valores do *makespan* médio para cada instância, nota-se a ocorrência de *outliers* em 73 das 110 instâncias, no entanto a quantidade é baixa variando de 1 a 5, sendo 1 *outliers* em 36 instâncias, 4 e 5 *outliers*, respectivamente em 5 e 4 instâncias e 28 instâncias possuindo de dois a três *outliers*. Neste estudo, esses *outliers* foram considerados como variações estocásticas naturais possíveis aos processos de produção, e consequentemente não receberam nenhum tipo de tratamento. Observa-se que há simetria em grande parte das instâncias, denotadas pela localização da mediana na porção mais central da caixa. A Tabela 13 descreve as estatísticas descritivas quanto a distribuição dos *makespans* médio utilizados para treinamento do modelo de Regressão Linear Múltipla. Quanto ao desempenho dos métodos desenvolvidos neste trabalho, estão expostos no Apêndice B - RESULTADOS TABELADOS.

A Tabela 14 exibe as métricas associadas ao treinamento do modelo de Regressão Linear Múltipla. Nota-se que os valores obtidos para o R^2 e o $R^2(VC)$, apresentaram valores superiores a 0,87 para todas as instâncias testadas, indicando que as variáveis dependentes do modelo são capazes de explicar mais de 87% dos *makespans* médios obtidos, com esses valores saltando para 0,99 nas instâncias com arranjos iguais ou superiores a 20 tarefas e 10 máquinas. A relação explicativa descrita era esperada, uma vez que as variáveis dependentes utilizadas para treinar o modelo, são utilizadas

diretamente para calcular o valor do *makespan* e por conseguinte o *makespan* médio. Contudo, essas métricas tão altas na maioria dos arranjos testados, sugerem que o modelo encontra-se com efeito direto da multicolinearidade decorrente da alta correlação entre as variáveis componentes do conjunto de dados de treinamento.

No que concerne aos erros calculados para o modelo, cujo os respectivas grandezas do EMA e da REMQ também são apresentados na Tabela 14. Nota-se que valores para os erros são relativamente baixos. Por exemplo, o REMQ obtido para o modelo de regressão referente a instância tai051 indica que o valor do *makespan* esperado previsto estaria com um erro de 3,315 unidades de tempo para mais ou para menos em relação ao valor esperado. O que não ocorreu nos testes direto dos valores analisados na predição equacional. Logo, analisando os valores desses erros de forma associada aos altos coeficientes de determinação, chega-se ao entendimento que o modelo pode está com *overfitting*, assim como em (Souza *et al.*, 2024).

Em relação ao Tempo de Treinamento do modelo RLM, este também encontra-se exposto na Tabela 14. De forma geral, nota-se que a maior parte das instâncias testadas demandam tempos relativamente curtos para consolidar o treinamento, apenas nos arranjos com dimensões de 200 tarefas e 20 máquinas os tempos ultrapassaram 200 segundos, enquanto nos demais casos o treinamento acontece em tempos inferiores a 40 segundos.

A Tabela 15, apresenta os resultados referentes ao método o MSNEHT-RLM-POOL, encontram-se na tabela os valores nominais dos *makespans* esperados previstos, chamado na tabela de Melhor *Makespan* (MM), o *Makespan* calculado pela simulação clássica (MMC), através da sequência associada ao MM, e o tempo de execução do método, o tempo de exploração das soluções armazenadas no *pool*, por meio das 500 simulações, designado na tabela por Tempo Total (TT(s)), Quantidades de Soluções no *pool* (QS) e o Número de Iterações (Nº It.) executadas. Do mesmo modo, a Tabela 16 expõe os resultados obtidos para o método GRASP-RLM-POOL, este análogo ao MSNEHT-RLM-POOL.

Ademais, a Tabela 18 mostram os resultados obtidos para os métodos híbridos que unem o aprendizado por reforço a meta-heurística *multi-start* aos métodos IQL-S E QLNEH-S. Enquanto a Tabela 20, exibe os resultado referente aos métodos NEHTBLMED e GRASPMED, estes realizam a previsão do *makespan* estimado sem o uso de simulações, ou seja, considerando apenas os tempos médios das instâncias testadas. Já a Tabela 17 representam os métodos simheurísticos comparativos, respectivamente, MSNEHT-SIMH e GRASP-SIMH. Por fim, a Tabela 19 apresentam os resultados relativos aos métodos baseados no algoritmo de aprendizagem por reforço QL, respectivamente o QLNEH-S e IQL-S.

Objetivando agregar maior robustez quanto a análise dos valores do MM encontrados para cada instância, foi conduzida uma análise estatística, com auxílio do *software* JAMOVI. Antes de definir o teste, foi necessário verificar se a distribuição amostral atendia ao critério de normalidade, então realizou-se o teste de Shapiro-Wilk com intuito de verificar a normalidade entre os valores dos MM obtidos pelos métodos explorados no estudo.

Após a execução do teste a um nível de significância $\alpha = 0,05$, obteve-se valores-p menores que a 0,05. Logo, os dados analisados não seguem uma distribuição normal, sendo necessário submeter a um teste não-paramétrico a análise das variações entre os métodos. Neste caso, foi escolhido o Teste de Friedman. A Tabela 4 mostra os valores encontrados para estatística W e o valor-p de Shapiro-Wilk para cada método.

Tabela 4 – Medidas da distribuição e Teste de Normalidade referente aos valores do MM para os métodos analisados

Método	**Mediana**	**W de Shapiro-Wilk**	**p de Shapiro-Wilk**	**25% percentil**	**50% percentil**	**75% percentil**
QLNEH-S	3963	0,857	$< 0,001$	2364	3963	6648
MSNEHT-SIMH	3898	0,856	$< 0,001$	2340	3898	6603
MSIQL-S	3941	0,857	$< 0,001$	2341	3941	6645
MSQLNEH-S	3933	0,857	$< 0,001$	2354	3933	6656
IQL-S	3957	0,857	$< 0,001$	2354	3957	6659
NEHTBLMED	3929	0,857	$< 0,001$	2396	3929	6555
MSNEHT-RLM-POOL	3938	0,857	$< 0,001$	2381	3938	6576
GRASPMED	4680	0,856	$< 0,001$	2578	4680	7825
GRASP-SIMH	4629	0,854	$< 0,001$	2614	4629	7917
GRASP-RLM-POOL	4723	0,858	$< 0,001$	2655	4723	7897

Fonte: Elaborada pelo autor (2025)

A Tabela 5 apresenta os resultados para o Teste de Friedman, observa-se que o valor-p foi inferior a 0,001. Logo a um nível de significância de $\alpha = 0,05$ podemos afirmar que há uma diferença estatisticamente significativa entre os valores do MM para os métodos analisados. Contudo, o Teste de Friedman não indica em quais métodos ocorreram está diferença. Diante disso, prosseguindo-se nas análises realizou-se a aplicação de um teste *Post-hoc* de comparações múltiplas pareadas para dados não paramétricos de Durbin-Conover, disponível no JAMOVI, visando verificar em quais métodos ocorreram tais diferenças. Os resultados estão dispostos na Tabela 6, podemos identificar que não houve diferenças estatisticamente significativa entre os seguintes pares de métodos:

- QLNEH-S e MSQLNEH-S, com um valor-p de 0,629;
- MSIQL-S e MSQLNEH-S, cujo o valor-p de 0,445;
- MSQLNEH-S e MSNEHT-RLM-POOL, cujo valor-p foi de 0,687;

- GRASPSIMH e GRASP-RLM-POOL, com valor-p de 0,159.

Consequentemente, os métodos supracitados não possuem diferenças estatisticamente significativa em relação aos valores preditos para o *makespan* esperado, observando os pares indicados. Em outras palavras, um método é equivalente ao par na predição. Contudo, em relação aos demais pares de métodos, o teste demonstrou que há diferenças estatisticamente significativas entre os modelos, ao nível de significância de $\alpha = 0,05$, ou seja, eles possuem diferenças em relação ao valor previsto para o $C_{\text{máx}}$ por cada método.

Tabela 5 – Teste Friedman Referente Aos Valores do MM Para os Métodos Analisados

χ^2	**gl**	**p**
690	9	$< 0,001$

Fonte: Elaborada pelo autor (2025)

Tabela 6 – Teste de Comparação *post-hoc* Durbin-Conover para o MM dos métodos desenvolvidos

Comparações		**Estatística**	**p**
QLNEH-S	MSNEHT-SIMH	10,908	$< 0,001$
—	MSIQL-S	4,549	$< 0,001$
—	MSQLNEH-S	3,784	$< 0,001$
—	IQL-S	0,483	0,629
—	NEHTBLMED	6,642	$< 0,001$
—	MSNEHT-RLM-POOL	4,186	$< 0,001$
—	GRASPMED	15,940	$< 0,001$
—	GRASP-SIMH	18,194	$< 0,001$
—	GRASP-RLM-POOL	19,603	$< 0,001$
MSNEHT-SIMH	MSIQL-S	6,360	$< 0,001$
—	MSQLNEH-S	7,125	$< 0,001$
—	IQL-S	11,391	$< 0,001$
—	NEHTBLMED	4,267	$< 0,001$
—	MSNEHT-RLM-POOL	6,722	$< 0,001$
—	GRASPMED	26,848	$< 0,001$
—	GRASP-SIMH	29,102	$< 0,001$
—	GRASP-RLM-POOL	30,511	$< 0,001$
MSIQL-S	MSQLNEH-S	0,765	0,445
—	IQL-S	5,032	$< 0,001$

Continua na página seguinte

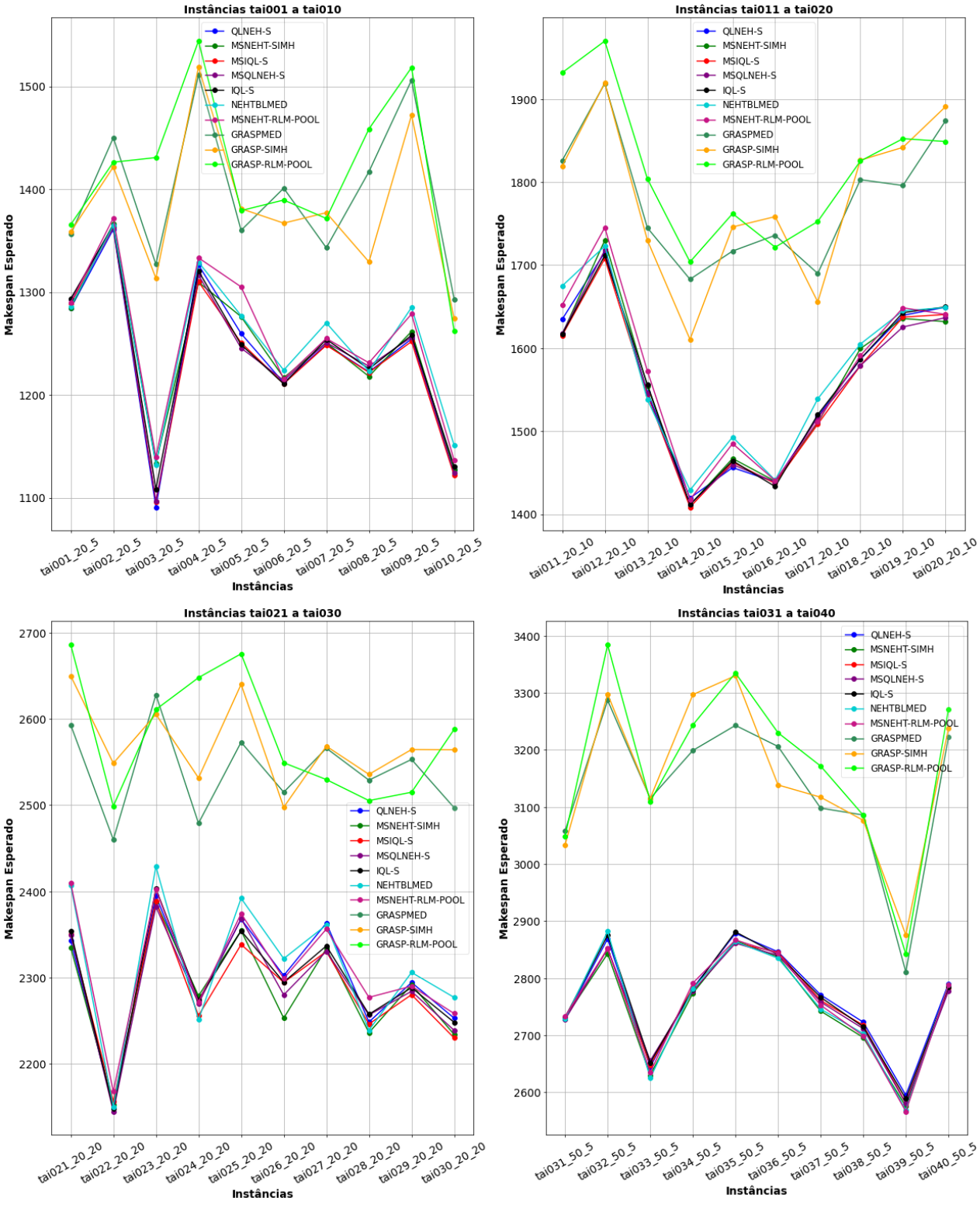
Tabela 6 – Teste de Comparação *post-hoc* Durbin-Conover para o MM dos métodos desenvolvidos (Continuação)

Comparações		**Estatística**	**p**
—	NEHTBLMED	2,093	0,037
—	MSNEHT-RLM-POOL	0,362	0,717
—	GRASPMED	20,488	$< 0,001$
—	GRASP-SIMH	22,743	$< 0,001$
—	GRASP-RLM-POOL	24,151	$< 0,001$
MSQLNEH-S	IQL-S	4,267	$< 0,001$
—	NEHTBLMED	2,858	0,004
—	MSNEHT-RLM-POOL	0,403	0,687
—	GRASPMED	19,724	$< 0,001$
—	GRASP-SIMH	21,978	$< 0,001$
—	GRASP-RLM-POOL	23,387	$< 0,001$
IQL-S	NEHTBLMED	7,125	$< 0,001$
—	MSNEHT-RLM-POOL	4,669	$< 0,001$
—	GRASPMED	15,457	$< 0,001$
—	GRASP-SIMH	17,711	$< 0,001$
—	GRASP-RLM-POOL	19,120	$< 0,001$
NEHTBLMED	MSNEHT-RLM-POOL	2,455	0,014
—	GRASPMED	22,582	$< 0,001$
—	GRASP-SIMH	24,836	$< 0,001$
—	GRASP-RLM-POOL	26,245	$< 0,001$
MSNEHT-RLM-POOL	GRASPMED	20,126	$< 0,001$
—	GRASP-SIMH	22,380	$< 0,001$
—	GRASP-RLM-POOL	23,789	$< 0,001$
GRASPMED	GRASP-SIMH	2,254	0,024
—	GRASP-RLM-POOL	3,663	$< 0,001$
GRASP-SIMH	GRASP-RLM-POOL	1,409	0,159

Fonte: Elaborada pelo autor (2025)

Realizando o teste de normalidade de Shapiro-Wilk para o MMC ao $\alpha = 0,05$, constata-se que os dados também não seguem uma distribuição normal. Ou seja, também são dados não

Figura 20 – Valores do MM *makespan* estocástico estimado para cada método implementado para as instâncias de tai001 a tai040

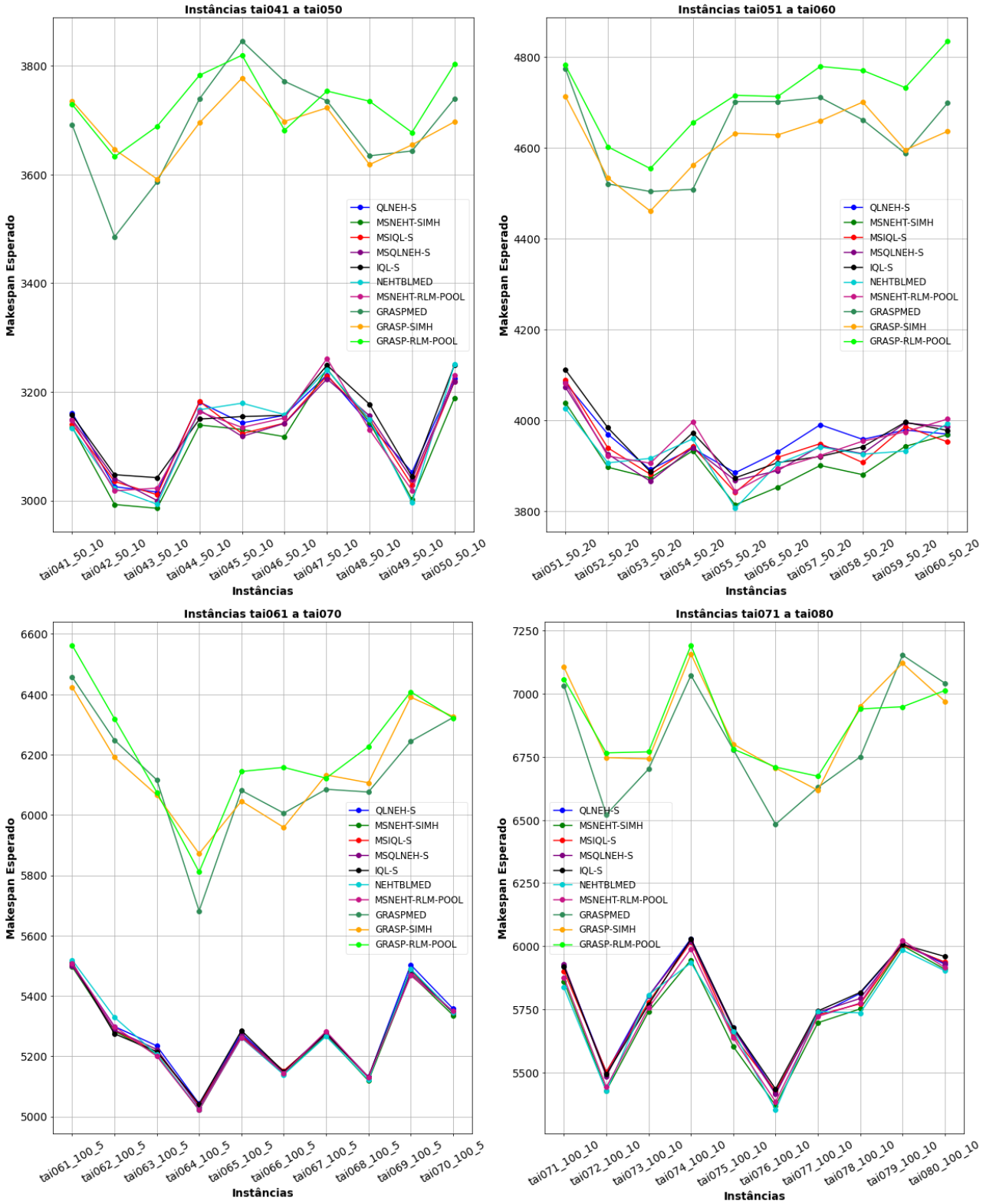


Fonte: Elaborada pelo autor (2025)

paramétricos. Assim, aplicou-se o Teste de Friedman, cujo resultado está exposto na Tabela 8 e demonstra que há diferenças estatisticamente significativas entre os métodos analisados.

Conduzindo o teste *post hoc* de Durbin-conover a $\alpha = 0{,}05$, cujos os resultados estão

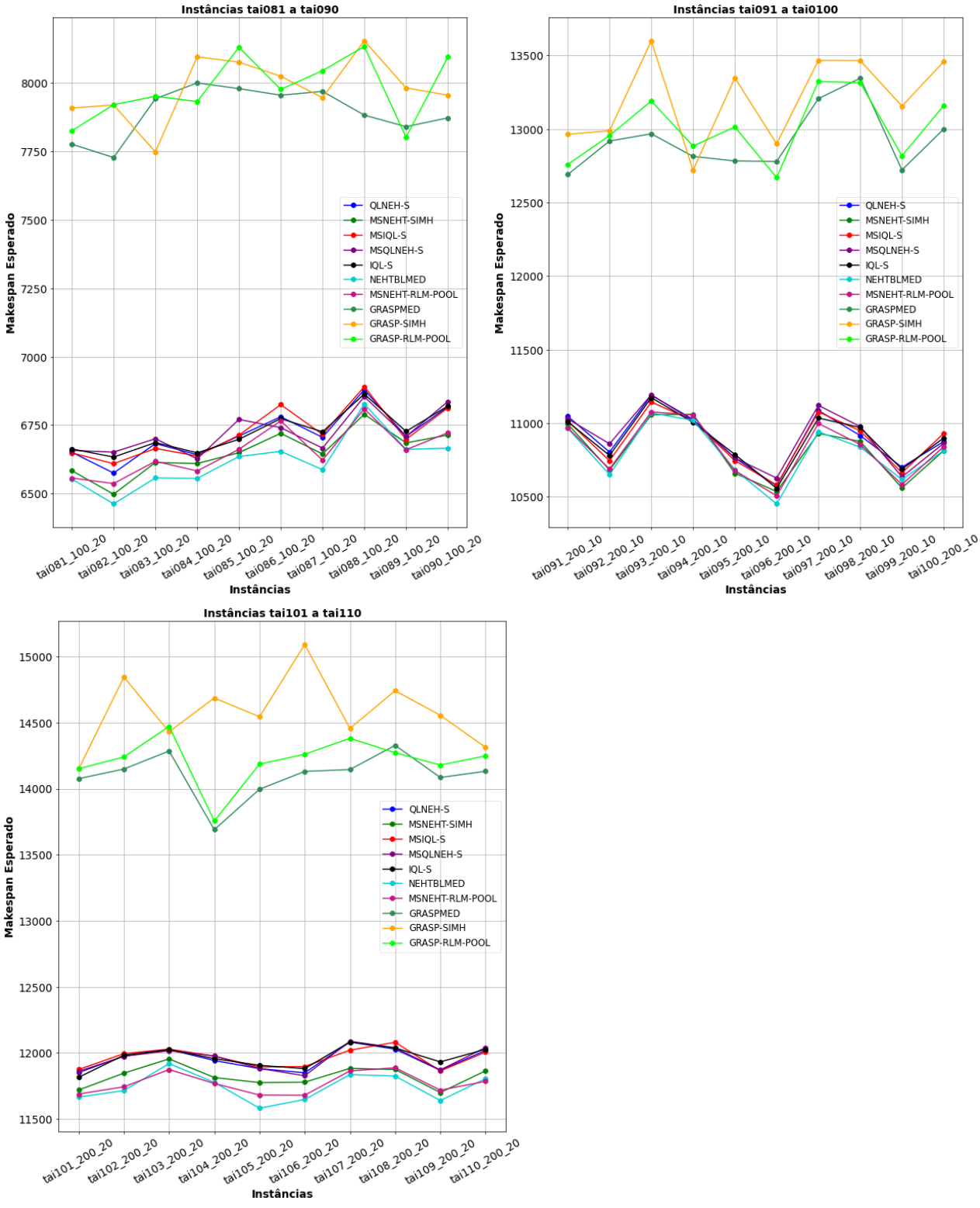
Figura 21 – Valores do MM *makespan* estocástico estimado para cada método implementado para as instâncias de tai041 a tai080



Fonte: Elaborada pelo autor (2025)

expostos na Tabela 9. Demonstra-se que há uma diferença estatisticamente significativa em relação ao MMC entre alguns métodos, excetuando-se os métodos que ao nível de significância de 0,05, apresentaram valor-p inferior a 0,05, os quais estão listados abaixo:

Figura 22 – Valores do MM *makespan* estocástico estimado para cada método implementado para as instâncias de tai081 a tai110



Fonte: Elaborada pelo autor (2025)

- QLNEH-S e MSIQLS-S, valor-p igual a 0,034;
- QLNEH-S e IQL-S, valor-p igual a 0,398;
- QLNEH-S e NEHTBLMED, valor-p igual a 0,421;

Tabela 7 – Medidas da distribuição e Teste de Normalidade referente aos valores do MMC para os métodos analisados

Método	**Mediana**	**W de Shapiro-Wilk**	**p de Shapiro-Wilk**	**25% percentil**	**50% percentil**	**75% percentil**
QLNEH-S	3965	0,858	$< 0,001$	2373	3965	6657
MSNEHT-SIMH	3916	0,857	$< 0,001$	2345	3916	6617
MSIQL-S	3968	0,858	$< 0,001$	2350	3968	6670
IQL-S	3960	0,857	$< 0,001$	2352	3960	6657
MSQLNEH-S	3961	0,857	$< 0,001$	2377	3961	6690
MSNEHT-RLM-POOL	3938	0,857	$< 0,001$	2382	3938	6576
NEHTBLMED	3961	0,858	$< 0,001$	2407	3961	6599
GRASPMED	4806	0,857	$< 0,001$	2564	4806	8007
GRASP-SIMH	4761	0,855	$< 0,001$	2750	4761	8073
GRASP-RLM-POOL	4724	0,858	$< 0,001$	2655	4724	7897

Fonte: Elaborada pelo autor (2025)

Tabela 8 – Teste Friedman para o MMC

χ^2	**gl**	**p**
690	9	$< 0,001$

Fonte: Elaborada pelo autor (2025)

- MSIQLS-S e IQL-S, valor-p igual a 0,07;
- MSNEHT-SIMH e MSNEHT-RLM-POOL, valor-p igual a 0,840;
- IQL-S e NEHTBLMED, valor-p igual a 0,099;
- GRASPMED e GRASP-SIMH, valor-p igual a 0,184;

Tabela 9 – Teste de Comparação *post-hoc* Durbin-Conover para o MMC dos métodos desenvolvidos

Comparações		**Estatística**	**p**
QLNEH-S	MSNEHT-SIMH	6,284	$< 0,001$
—	MSIQL-S	0,967	0,334
—	MSQLNEH-S	2,981	0,003
—	IQL-S	0,846	0,398
—	NEHTBLMED	0,806	0,421
—	MSNEHT-RLM-POOL	6,082	$< 0,001$
—	GRASPMED	22,114	$< 0,001$
—	GRASP-SIMH	23,443	$< 0,001$
—	GRASP-RLM-POOL	17,280	$< 0,001$
MSNEHT-SIMH	MSIQL-S	7,251	$< 0,001$
—	MSQLNEH-S	9,265	$< 0,001$
—	IQL-S	5,438	$< 0,001$

Continua na página seguinte

Tabela 9 – Teste de Comparação *post-hoc* Durbin-Conover para o MMC dos métodos desenvolvidos (Continuação)

Comparações		**Estatística**	**p**
—	NEHTBLMED	7,089	$< 0{,}001$
—	MSNEHT-RLM-POOL	0,201	0,840
—	GRASPMED	28,398	$< 0{,}001$
—	GRASP-SIMH	29,727	$< 0{,}001$
—	GRASP-RLM-POOL	23,564	$< 0{,}001$
MSIQL-S	MSQLNEH-S	2,014	0,044
—	IQL-S	1,813	0,070
—	NEHTBLMED	0,161	0,872
—	MSNEHT-RLM-POOL	7,049	$< 0{,}001$
—	GRASPMED	21,147	$< 0{,}001$
—	GRASP-SIMH	22,477	$< 0{,}001$
—	GRASP-RLM-POOL	16,314	$< 0{,}001$
MSQLNEH-S	IQL-S	3,827	$< 0{,}001$
—	NEHTBLMED	2,175	0,030
—	MSNEHT-RLM-POOL	9,063	$< 0{,}001$
—	GRASPMED	19,133	$< 0{,}001$
—	GRASP-SIMH	20,463	$< 0{,}001$
—	GRASP-RLM-POOL	14,300	$< 0{,}001$
IQL-S	NEHTBLMED	1,652	0,099
—	MSNEHT-RLM-POOL	5,236	$< 0{,}001$
—	GRASPMED	22,960	$< 0{,}001$
—	GRASP-SIMH	24,289	$< 0{,}001$
—	GRASP-RLM-POOL	18,126	$< 0{,}001$
NEHTBLMED	MSNEHT-RLM-POOL	6,888	$< 0{,}001$
—	GRASPMED	21,309	$< 0{,}001$
—	GRASP-SIMH	22,638	$< 0{,}001$
—	GRASP-RLM-POOL	16,475	$< 0{,}001$
MSNEHT-RLM-POOL	GRASPMED	28,197	$< 0{,}001$
—	GRASP-SIMH	29,526	$< 0{,}001$

Continua na página seguinte

Tabela 9 – Teste de Comparação *post-hoc* Durbin-Conover para o MMC dos métodos desenvolvidos (Continuação)

Comparações		**Estatística**	**p**
—	GRASP-RLM-POOL	23,363	$< 0{,}001$
GRASPMED	GRASP-SIMH	1,329	0,184
—	GRASP-RLM-POOL	4,834	$< 0{,}001$
GRASP-SIMH	GRASP-RLM-POOL	6,163	$< 0{,}001$

Fonte: Elaborada pelo autor (2025)

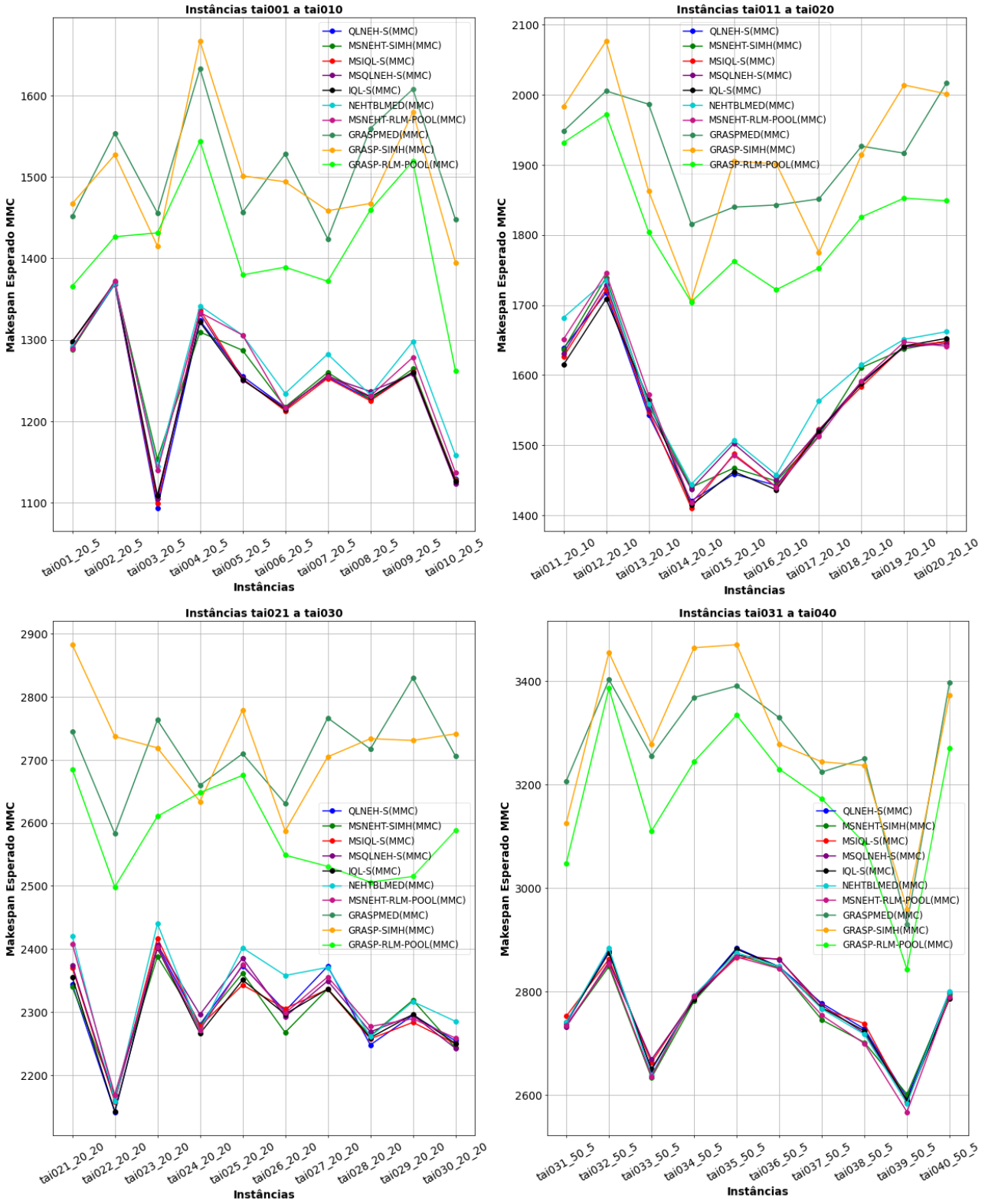
As Figuras 20, 21 e 22 apresentam o graficamente o MM encontrado por cada método em cada instância testada. Nota-se que os métodos baseados na meta-heurística GRASP tiveram um desempenho preditivo inferior em relação aos demais métodos testados. Comparativamente aos outros métodos híbridos implementados a diferença entre a GRASP e os métodos que adotaram a meta-heurística *Multi-start* reside apenas na forma de solução construtiva, enquanto o GRASP usa a técnica baseada na Lista de Candidatos Restrita, as demais utilizam a heurística NEHT, ressaltando que a tarefa selecionada é sempre colocada no final da sequência, enquanto a lógica do NEHT escolhe a posição que minimiza o *makespan*, ou os métodos baseados nos algoritmos de ML pela técnica de aprendizagem por reforço. Além disso, os valores nominais dos outros métodos estão todos dispostos em uma mesma área, indicando influência direta da solução construtiva no desempenho dos métodos.

As Figuras 23, 24 e 25 mostram graficamente os resultados associados as 10.000 SMC, na forma clássica, para a sequência associada ao MM encontrado por cada um dos métodos, denotando-se por MMC. Identifica-se, a mesma tendência para os valores esperados simulados, em relação aos valores de MM. Entretanto, há amplitudes maiores nos campos que utilizar a metodologia GRASP, principalmente em instâncias pequenas e amplitude menores para os demais métodos. Vale ressaltar que adoção da medida de verificação por meio da simulação longa que resultou no valor do MMC se deu em razão da precisão associada ao valor gerado para uma sequência eleita.

Por haver diferenças estatisticamente significativas, entre os vários métodos e a influência da construtiva no desempenho foi necessário examinar individualmente o resultado para cada instância no espectro dos métodos em comparação. Esses gráficos encontram-se disponíveis no repositório: https://github.com/brunol3/Resultados-por-instancia/tree/main.

Dentre os métodos explorados, nenhum deles conseguiu apresentar o melhor desempenho absoluto em todas as instâncias. A Figura 26 mostra a frequência com a qual os métodos pontuaram o melhor desempenho em relação ao MM, observando as dimensões do arranjo: quantidade de tarefas e número de máquinas. Observa-se que nas instâncias menores (dimensão 20 tarefas), os métodos

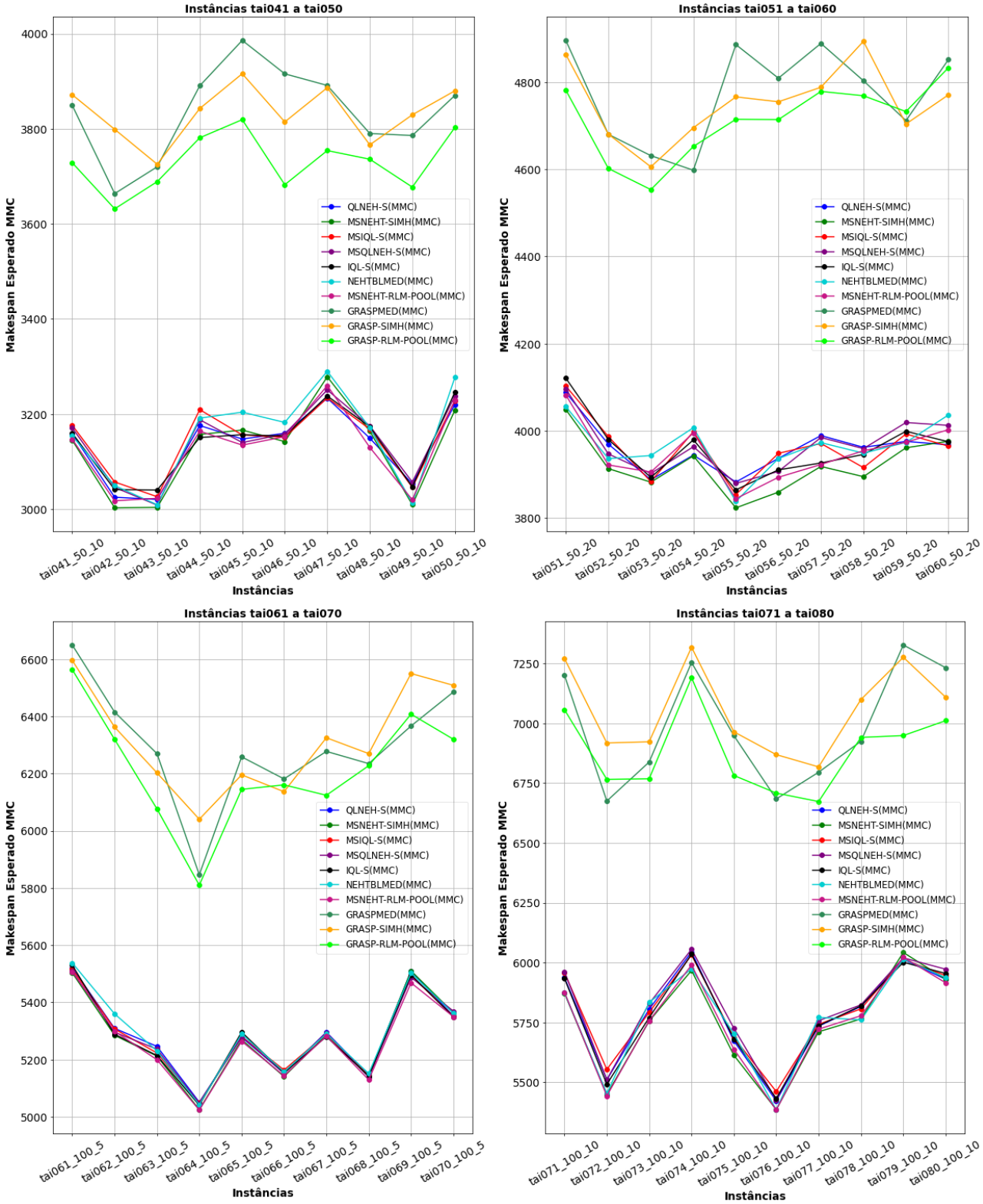
Figura 23 – Valores do MMC *makespan* estocástico estimado para cada método implementado para as instâncias de tai001 a tai040



Fonte: Elaborada pelo autor (2025)

QLNEH-S e MSIQLS-S demonstraram maior frequência em convergir para um valor menor valor do $C_{máx}$ esperado em comparação aos outros métodos. Nas instâncias intermediárias que variam de 50 tarefas e 5 máquinas até 100 tarefas e 5 máquinas a maior frequência convergência exitosa,

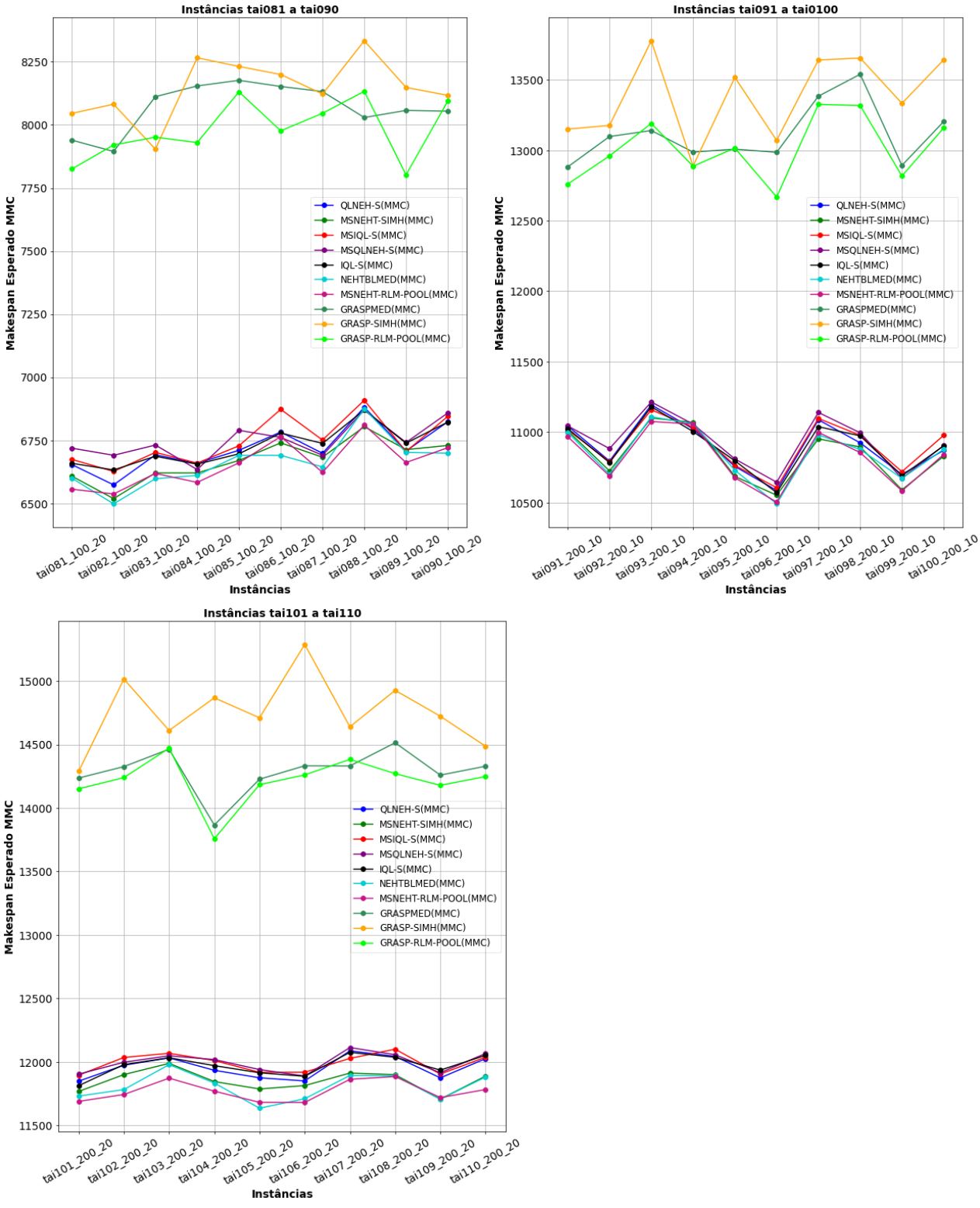
Figura 24 – Valores do MMC *makespan* estocástico estimado para cada método implementado para as instâncias de tai041 a tai080



Fonte: Elaborada pelo autor (2025)

ocorreu na simheurística MSNEHT-SIMH. Já nas instâncias maiores, cujo arranjo varia de 100 tarefas e 10 máquinas a 200 tarefas e 20 máquinas o método que demonstrou o melhor resultado foi o NEHTBLMED, o que pode ter ocorrido em razão da não realização das simulações, haja vista que o

Figura 25 – Valores do MMC *makespan* estocástico estimado para cada método implementado para as instâncias de tai081 a tai110

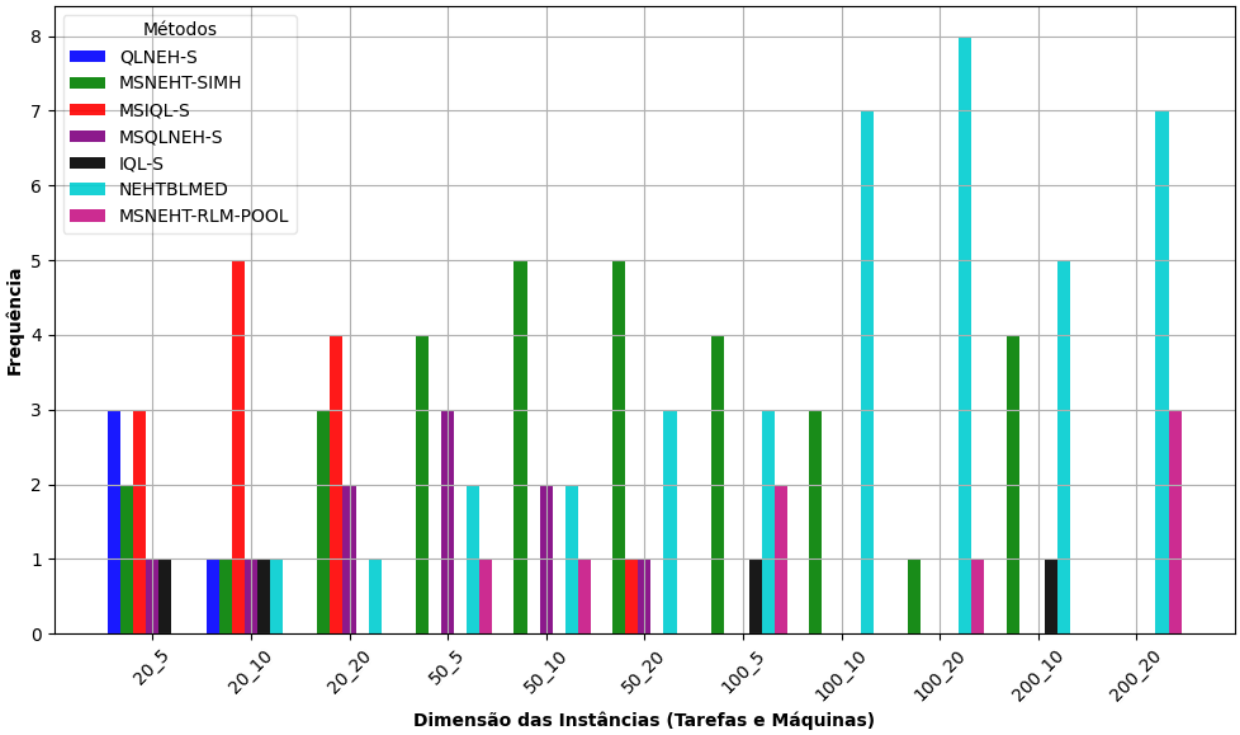


Fonte: Elaborada pelo autor (2025)

método pode usar todo tempo limite disponível para convergir, isto não ocorre nos métodos que usam a simulação, pois nestes o tempo limite é alcançado antes da finalização da busca local.

Outro ponto importante, foi a remoção dos métodos baseados no GRASP da Figura 26,

Figura 26 – Frequência do Melhor Desempenho dos Método em Relação a Dimensão das Instâncias



Fonte: Elaborada pelo autor (2025)

pois não obteve-se bons resultados em nenhum cenário. Diante do exposto, infere-se que embora haja alguma relação entre a dimensão da instância e os melhores métodos a serem utilizados, considerando-se as diferenças apresentadas entre as instâncias unitárias, principalmente nos arranjos menores que estão na mesma dimensão. Logo, interpreta-se que o método a ser recomendado é dependente tanto das dimensões quanto dos tempos de processamento da instância.

Os tempos de execução dos métodos também foram submetidos a uma avaliação estatística com intuito de verificar se há diferenças estatisticamente significativas em relação aos métodos desenvolvidos. Vale ressaltar que os modelos híbridos integrados ao modelo de aprendizagem supervisionada de RLM, o tempo considerado foi o TT, haja vista que a exploração das melhores soluções através do *pool* estão contempladas diretamente a resposta preditiva final dos métodos MSNEHT-RLM-POOL e GRASP-RLM-POOL. Assim como na avaliação dos MM foi realizado o teste de normalidade de Shapiro-Wilk a um nível de significância de $\alpha = 0,05$, com fins de prosseguir com o teste estatístico adequado. A Tabela 10 demonstra os valores referentes ao teste de Shapiro-Wilk, observa-se que em todos os métodos o valor-p apresentou a valores inferiores a 0,001. Logo, a $\alpha = 0,05$ podemos considerar que os dados não são normalmente distribuídos e deve-se aplicar um teste não-paramétrico, que neste caso, também foi o Teste de Friedman.

A Tabela 11 apresenta os resultados do Teste de Friedman relacionados aos tempos de

execução dos métodos desenvolvidos. Considerando um nível de significância de $\alpha = 0,05$, observa-se que o valor-p para o Teste de Friedman foi inferior a 0,001. Portanto, podemos afirmar que há uma diferença estatisticamente significativa entre os tempos de execução dos métodos implementados. Entretanto, é necessário identificar entre quais métodos se verificaram essas diferenças, sendo que, para isso, foi aplicado o teste *post-hoc* Durbin-Conover para comparação par a par dos métodos, os quais estão expostos na Tabela 12 também adotando um nível de significância de $\alpha = 0,05$, constatou-se uma diferença estatisticamente significativa em quase todos os pares de métodos, em termos dos tempos de execução, com exceção dos seguintes pares de algoritmos:

- MSNEHT-SIMH e MSIQLS-S, com um valor-p de 0,127;
- MSNEHT-SIMH e GRASP-RLM-POOL, cujo valor-p foi de 0,432;
- MSIQLS-S e MSQLNEH-S, com valor-p de 0,518.

Denotando-se que não há diferenças consideráveis entre os métodos supramencionados. Os gráficos que demonstram o desempenho de cada instância em relação aos tempos de execução estão disponíveis em: https://github.com/brunol3/Resultados-por-instancia/tree/main/graficos_tempos. Nota-se que os métodos que envolvem a estimativa pelos tempos médios (NEHTBLMED e GRASPMED) apresentaram os menores tempos de execução em todas as instâncias, isso está coerente, pois em razão desses métodos não realizar nenhum tipo de simulação, naturalmente tende a serem mais rápidos. Destacando, ainda que o NEHTBLMED executa apenas uma vez. Sendo assim, o GRASPMED ainda possui uma vantagem, pois executa 10 iterações. De modo geral, os métodos QLNEH-S, IQL-S e MSNEHT-RLM-POOL estão em seguida como métodos mais eficientes. No entanto, ressalta-se que estes tempos estão relacionados a parametrização dos métodos, no caso da parametrização escolhida, verifica-se que entre os demais métodos os tempos de execução para se obter uma solução não é impeditivo ao uso dos métodos aqui abordados.

Tabela 12 – Teste de Comparação *post-hoc* Durbin-Conover para os Tempos de Execução dos métodos desenvolvidos

Comparações		**Estatística**	**p**
QLNEH-S	MSNEHT-SIMH	13,500	$< 0,001$
—	MSIQL-S	11,974	$< 0,001$
—	MSQLNEH-S	11,327	$< 0,001$
—	IQL-S	9,894	$< 0,001$
—	NEHTBLMED	22,469	$< 0,001$
—	MSNEHT-RLM-POOL	4,808	$< 0,001$

Continua na página seguinte

Tabela 12 – Teste de Comparação *post-hoc* Durbin-Conover para os Tempos de Execução dos métodos desenvolvidos (Continuação)

Comparações		**Estatística**	**p**
—	GRASPMED	17,384	$< 0,001$
—	GRASP-SIMH	7,629	$< 0,001$
—	GRASP-RLM-POOL	14,286	$< 0,001$
MSNEHT-SIMH	MSIQL-S	1,526	0,127
—	MSQLNEH-S	2,173	0,030
—	IQL-S	23,394	$< 0,001$
—	NEHTBLMED	35,970	$< 0,001$
—	MSNEHT-RLM-POOL	18,308	$< 0,001$
—	GRASPMED	30,884	$< 0,001$
—	GRASP-SIMH	5,872	$< 0,001$
—	GRASP-RLM-POOL	0,786	0,432
MSIQL-S	MSQLNEH-S	0,647	0,518
—	IQL-S	21,868	$< 0,001$
—	NEHTBLMED	34,444	$< 0,001$
—	MSNEHT-RLM-POOL	16,783	$< 0,001$
—	GRASPMED	29,358	$< 0,001$
—	GRASP-SIMH	4,346	$< 0,001$
—	GRASP-RLM-POOL	2,312	0,021
MSQLNEH-S	IQL-S	21,221	$< 0,001$
—	NEHTBLMED	33,797	$< 0,001$
—	MSNEHT-RLM-POOL	16,135	$< 0,001$
—	GRASPMED	28,711	$< 0,001$
—	GRASP-SIMH	3,699	$< 0,001$
—	GRASP-RLM-POOL	2,959	0,003
IQL-S	NEHTBLMED	12,576	$< 0,001$
—	MSNEHT-RLM-POOL	5,086	$< 0,001$
—	GRASPMED	7,490	$< 0,001$
—	GRASP-SIMH	17,522	$< 0,001$
—	GRASP-RLM-POOL	24,180	$< 0,001$

Continua na página seguinte

Tabela 12 – Teste de Comparação *post-hoc* Durbin-Conover para os Tempos de Execução dos métodos desenvolvidos (Continuação)

Comparações		**Estatística**	**p**
NEHTBLMED	MSNEHT-RLM-POOL	17,661	$< 0,001$
—	GRASPMED	5,086	$< 0,001$
—	GRASP-SIMH	30,098	$< 0,001$
—	GRASP-RLM-POOL	36,756	$< 0,001$
MSNEHT-RLM-POOL	GRASPMED	12,576	$< 0,001$
—	GRASP-SIMH	12,437	$< 0,001$
—	GRASP-RLM-POOL	19,094	$< 0,001$
GRASPMED	GRASP-SIMH	25,012	$< 0,001$
—	GRASP-RLM-POOL	31,670	$< 0,001$
GRASP-SIMH	GRASP-RLM-POOL	6,658	$< 0,001$

Fonte: Elaborada pelo autor (2025)

6.1 SIMULADOR

O simulador WEB foi desenvolvido na linguagem de programação Python, utilizando a biblioteca *streamlit*. O *software* requer a inserção dos tempos de processamento de cada tarefa *i* em cada máquina *j*, nos formato txt ou csv. Em seguida, o usuário pode definir o modelo utilizado para sua análise ou estimação do *makespan* esperado para o arranjo definido. Isso ocorre, nas possibilidades dos métodos desenvolvidos nesta dissertação e foram codificados no simulador para serem utilizados a critério do usuário.

A incorporação dos métodos implementados com exceção dos modelos que utilizam o GRASP, devido ao baixo desempenho frente a todos os outros métodos, e aos modelos baseados na MSNEHT-RLM-POOL, este em virtude da dinâmica exigir a criação de um conjunto de dados para treinamento o que demanda um esforço computacional adicional, bem como o fato do teste estatístico ter indicado que o MSQLNEH-S e o MSNEHT-RLM-POOL não possuem diferenças estatisticamente significativa, isto implica que um método pode ser utilizado em vez do outro. Assim os módulos do simulador desenvolvido, estão demonstrados na Figura 27.

A Figura 28 mostra a Tela de Login do Simulador, a Figura 29 apresenta a tela inicial do sistema. Enquanto a Figura 30 o menu das simheurísticas híbridas, a Figura 31 ilustra o menu dos métodos de aprendizagem por reforço, implementados neste trabalho. A Figura 32 exibe a execução

Tabela 10 – Medidas da distribuição e Teste de Normalidade referente aos Tempos de Execução para os métodos analisados

Método	**Mediana**	**W de Shapiro-Wilk**	**p de Shapiro-Wilk**	**25% Percentil**	**50% Percentil**	**75% Percentil**
QLNEH-S	128	0,585	$< 0,001$	22,40	128	700
MSNEHT-SIMH	1525	0,773	$< 0,001$	86,70	1525	3600
MSQLNEH-S	1458	0,771	$< 0,001$	89,20	1458	3600
MSIQL-S	1463	0,772	$< 0,001$	89,50	1463	3600
IQL-S	123	0,585	$< 0,001$	22,00	123	674
NEHTBLMED	0,740	0,563	$< 0,001$	0,05	0,740	5,73
MSNEHT-RLM-POOL	124	0,585	$< 0,001$	22,10	124	680
GRASPMED	7,11	0,564	$< 0,001$	0,46	7,11	56,40
GRASP-SIMH	1510	0,773	$< 0,001$	84,50	1510	3600
GRASP-RLM-POOL	1517	0,804	$< 0,001$	85,00	1517	3657

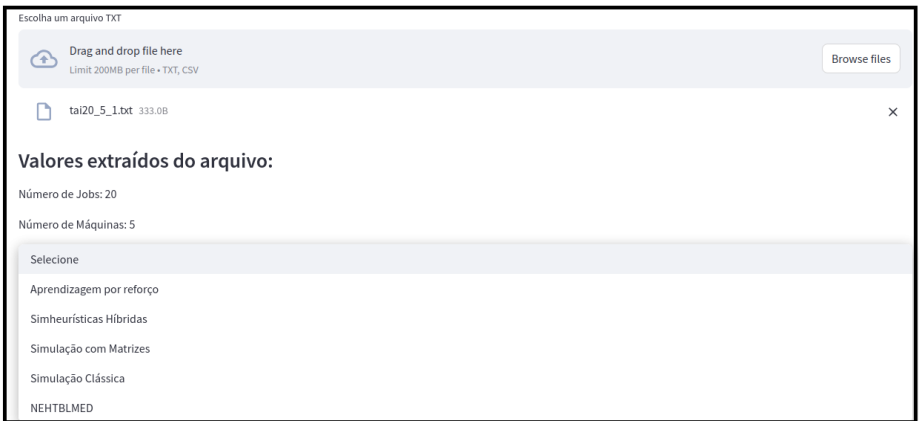
Fonte: Elaborada pelo autor (2025)

Tabela 11 – Teste Friedman referente aos Tempos de Execução para os métodos analisados

χ^2	**gl**	**p**
762	9	$< 0,001$

Fonte: Elaborada pelo autor (2025)

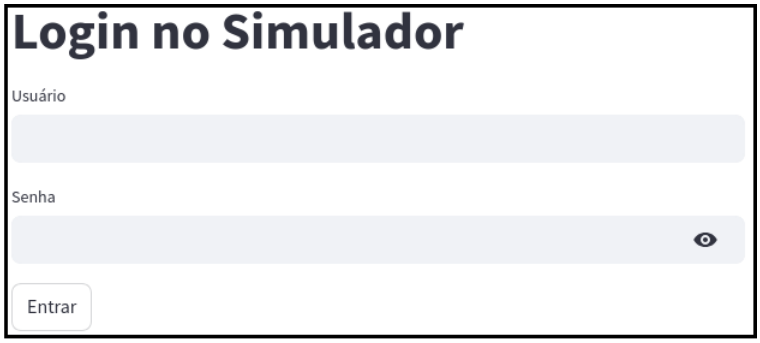
Figura 27 – Módulos do Simulador



Fonte: Elaborada pelo autor (2025)

do método NEHTBLMED e a Figura 33 mostra o gráfico de Gantt interativo disponível no simulador, plotando os tempos médios informados de acordo com ordenamento de uma programação de tarefas.

Figura 28 – Tela de Login do Simulador

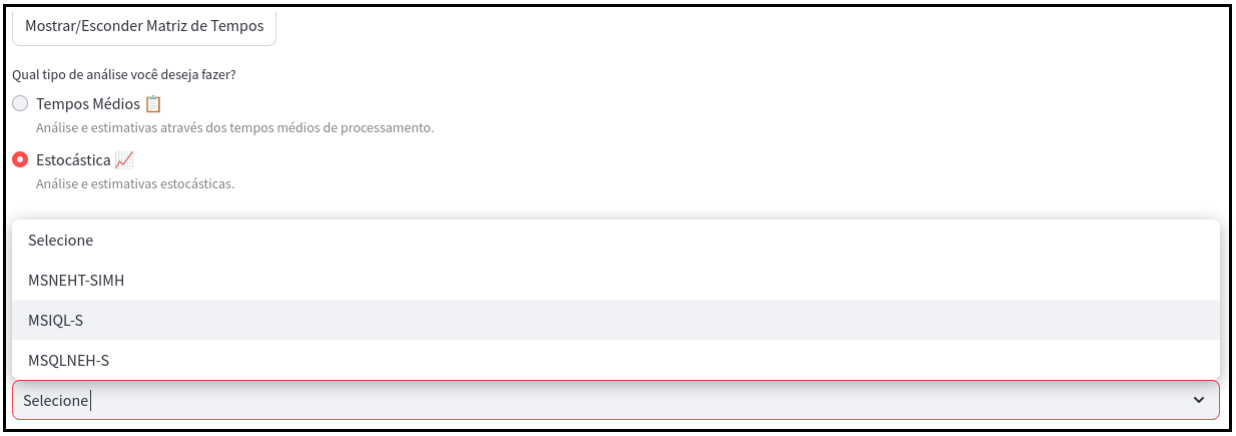


Fonte: Elaborada pelo autor (2025)

Figura 29 – Tela Inicial do Simulador

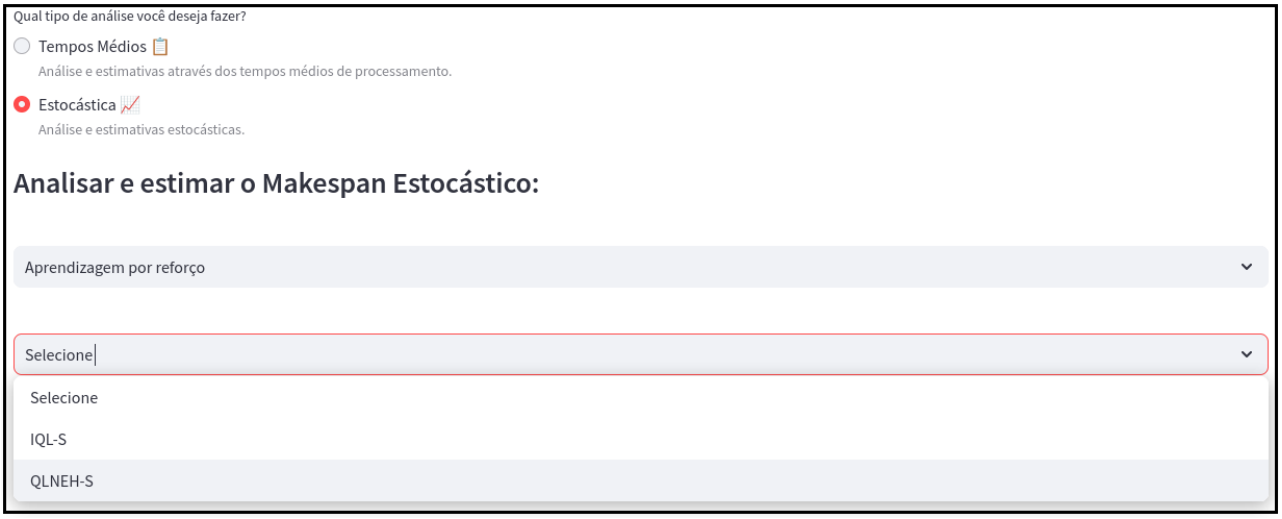
Fonte: Elaborada pelo autor (2025)

Figura 30 – Opções no Menu de Simheurísticas Híbridas



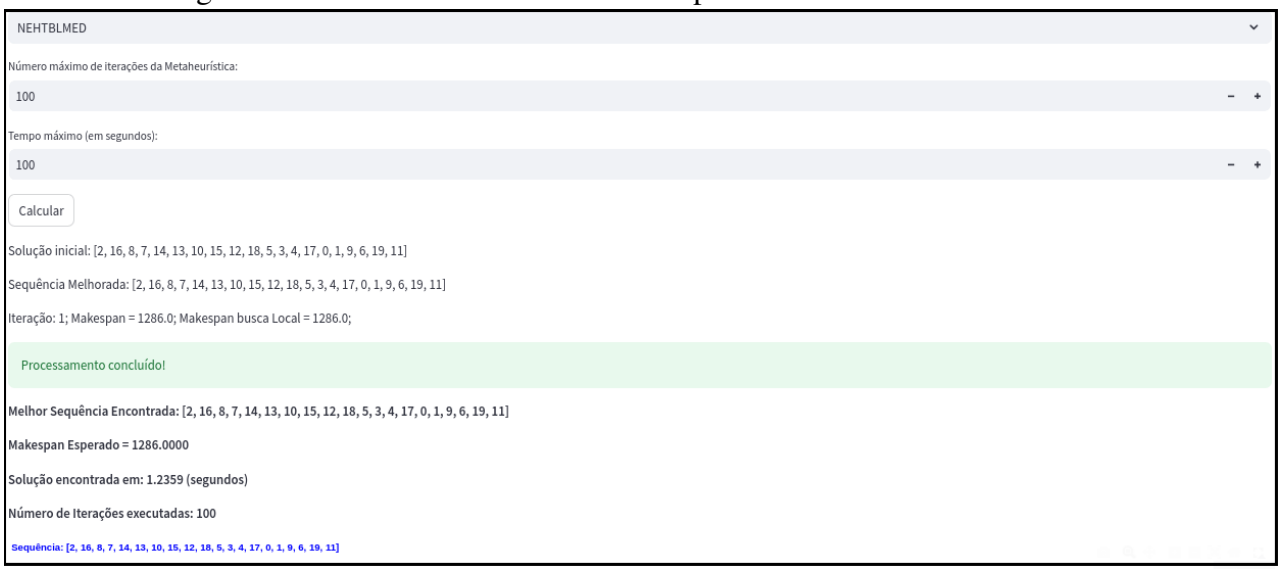
Fonte: Elaborada pelo autor (2025)

Figura 31 – Opções no Menu de Aprendizagem por reforço



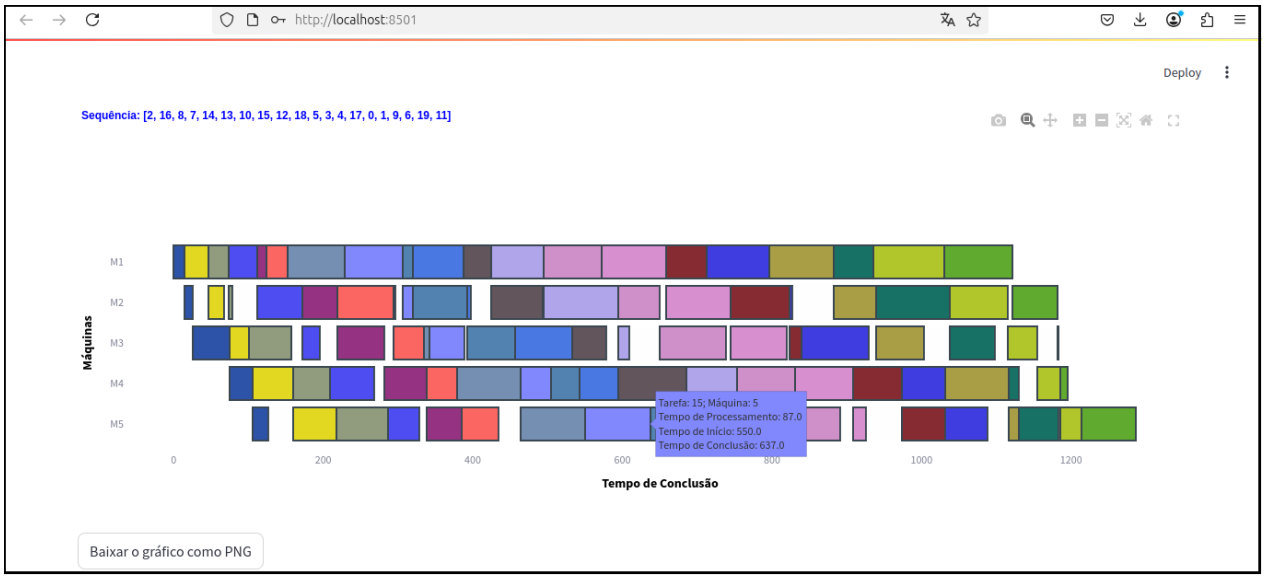
Fonte: Elaborada pelo autor (2025)

Figura 32 – Resultado de Uma Análise a partir do Método NEHTBLMED



Fonte: Elaborada pelo autor (2025)

Figura 33 – Gráfico de Gantt interativo do Simulador



Fonte: Elaborada pelo autor (2025)

7 CONSIDERAÇÕES FINAIS

As transformações tecnológicas e mercadológicas, aliadas aos aumentos do custo de produção exigem um planejamento bem definido. O que por sua vez, estimula a busca por estratégias capazes de tornar os sistemas produtivos mais eficientes. Assim, os métodos explorados nesta pesquisa auxiliam diretamente nesse processo, uma vez que muitos problemas associados a programação da produção são de caráter combinatório e difíceis de garantir uma solução ótima.

Neste trabalho, considerou-se a junção de técnicas de otimização com algoritmos de *machine learning* como forma de buscar melhores soluções para o sequenciamento da produção, considerando questões estocásticas simuladas através da simulação de Monte Carlo.

A pesquisa, propôs uma nova forma forma de calcular o *makespan* esperado utilizando a Simulação de Monte Carlo. E baseando-se em algoritmos e outros trabalhos da literatura desenvolveu 10 métodos diferentes com o objetivo de encontrar um melhor sequenciamento da produção sob a inserção de incerteza, através da estimação do *makespan*, visando minimiza-ló, foram implementados: dois métodos híbridos que utilizam modelos supervisionado de o modelo de Regressão Linear Múltipla com meta-heurística *multi-start* e a heurística construtiva NEHT; quatro métodos que envolveram aprendizado por reforço, todos baseados no algoritmo QL, sendo dois métodos adaptações de algoritmos da literatura utilizados no contexto do trabalho, e os outros dois meta-heurísticas híbridas que usam os dois algoritmos supracitados como estratégia construtiva; e dois métodos para fins comparativos, um que recorre a Simulação de Monte Carlo como base para geração e otimização estocástica e o outro que utiliza os tempos médios decorrentes de instâncias da literatura para estimar o *makespan*.

Nenhum método foi absoluto em minimizar o *makespan* esperado em todas as instâncias. Constatou-se que a solução construtiva impacta diretamente no desempenho do método. Diante disso, os métodos baseados na metaheurística GRASP com solução construtiva baseada na LCR apresentaram piores resultados em relação a todos os métodos explorados no trabalho. Atentando, que o NEHT coloca a tarefa na melhor posição, enquanto a LCR implementada sempre coloca a tarefa selecionada na última posição.

Através da análise estatística e gráfica identificou-se que mais adequado, dentre os desenvolvidos, dependem das dimensões associadas ao arranjo da instância e aos tempos de processamento.

Diante do exposto, foi desenvolvido um simulador WEB capaz de auxiliar diretamente a programação da produção. Este engloba os modelos explorados no trabalho, com exceção dos métodos que utilizam a meta-heurística GRASP e os modelos que usam a RLM.

O trabalho desenvolvido traz impactos significativos no âmbito econômico, uma vez que

pode auxiliar a encontrar uma programação da produção mais eficiente e consequentemente pode melhorar a utilização dos recursos, reduzir o tempo de ciclo da produção de produtos e promover um aumento da competitividade. No plano ambiental, a possibilidade de redução do consumo de energia apresenta-se como destaque, haja vista que a eficiência da programação da produção pode minimizar o gasto energético. Na perspectiva social, o trabalho contribuí no auxílio a tomada de decisão mais assertiva por parte da equipe gestora da produção de unidades fabris que possam utilizar os métodos desenvolvidos neste trabalho.

Como sugestão de trabalhos futuros, pode-se executar os métodos para uma quantidade maior de instâncias e empreender esforços na agregação de redes neurais artificiais para o aprendizado profundo como forma de melhorar as soluções encontradas. Haja vista, que ainda sim não podemos garantir a otimalidade das soluções encontradas. E buscar parcerias para teste em cenários fabris reais para comprovar a eficácia dos métodos em contextos aplicados.

REFERÊNCIAS

ALLAM, Z.; BIBRI, S. E.; SHARPE, S. A. The rising impacts of the covid-19 pandemic and the russia–ukraine war: Energy transition, climate justice, global inequality, and supplychain disruption. **Resources**, v. 11, n. 11, 2022. ISSN 2079-9276. Disponível em: https://www.mdpi.com/2079-9276/11/11/99.

AMARAL, H. F.; URRUTIA, S.; HVATTUM, L. M. Delayed improvement local search. **Journal of Heuristics**, Springer, v. 27, p. 923–950, 2021.

AMARASINGHE, K.; RODOLFA, K. T.; LAMBA, H.; GHANI, R. Explainable machine learning for public policy: Use cases, gaps, and research directions. **Data & Policy**, v. 5, p. e5, 2023.

AMIRGHASEMI, M. An effective decomposition-based stochastic algorithm for solving the permutation flow-shop scheduling problem. **Algorithms**, v. 14, n. 4, 2021. ISSN 1999-4893. Disponível em: https://www.mdpi.com/1999-4893/14/4/112.

AZAB, E.; NAFEA, M.; SHIHATA, L. A.; MASHALY, M. A machine-learning-assisted simulation approach for incorporating predictive maintenance in dynamic flow-shop scheduling. **Applied Sciences**, v. 11, n. 24, 2021. ISSN 2076-3417. Disponível em: https://www.mdpi.com/2076-3417/11/24/11725.

AZADEH, A.; MALEKI-SHOJA, B.; SHEIKHALISHAHI, M.; ESMAILI, A.; ZIAEIFAR, A.; MORADI, B. A simulation optimization approach for flow-shop scheduling problem: a canned fruit industry. **INTERNATIONAL JOURNAL OF ADVANCED MANUFACTURING TECHNOLOGY**, v. 77, n. 1-4, p. 751–761, MAR 2015. ISSN 0268-3768.

BARMAN, N.; JAMMEH, E.; GHORASHI, S. A.; MARTINI, M. G. No-reference video quality estimation based on machine learning for passive gaming video streaming applications. **IEEE Access**, v. 7, p. 74511–74527, 2019.

BENDA, F.; BRAUNE, R.; DOERNER, K. F.; HARTL, R. F. A machine learning approach for flow shop scheduling problems with alternative resources, sequence-dependent setup times, and blocking. **OR SPECTRUM**, v. 41, n. 4, SI, p. 871–893, DEC 2019. ISSN 0171-6468.

BERTSEKAS, D. P.; TSITSIKLIS, J. N. **Neuro-dynamic programming**. Massachusetts, USA: Athena Scientific, 1996. ISBN 1-886529-10-8.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. California, USA: Springer, 2006. ISBN 10: 0-387-31073-8.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 35, n. 3, p. 268–308, sep 2003. ISSN 0360-0300. Disponível em: https://doi.org/10.1145/937503.937505.

BOWDEN, R. O.; GOGG, T. J.; HARRELL, C. R. **Simulação De Sistemas: Aprimorando Processos de Logística, Serviços e Manufatura**. 5. ed. Rio de Janeiro, Brasil: Elsevier Brasil, 2013. ISBN 978-85-352-7162-1.

BREEDAM, A. V. Comparing descent heuristics and metaheuristics for the vehicle routing problem. **Computers & Operations Research**, v. 28, n. 4, p. 289–315, 2001. ISSN 0305-0548. Disponível em: https://www.sciencedirect.com/science/article/pii/S030505489900101X.

BROWN, A. P. G.; LOMNICKI, Z. A. Some applications of the "branch-and-bound" algorithm to the machine scheduling problem. **Journal of the Operational Research Society**, Taylor & Francis, v. 17, n. 2, p. 173–186, 1966. Disponível em: https://doi.org/10.1057/jors.1966.25.

CAMPBELL, H. G.; DUDEK, R. A.; SMITH, M. L. A heuristic algorithm for the n job, m machine sequencing problem. **Management Science**, v. 16, n. 10, p. B–630–B–637, 1970. Disponível em: https://doi.org/10.1287/mnsc.16.10.B630.

CASTANEDA, J.; MARTIN, X. A.; AMMOURIOVA, M.; PANADERO, J.; JUAN, A. A. A fuzzy simheuristic for the permutation flow shop problem under stochastic and fuzzy uncertainty. **Mathematics**, v. 10, n. 10, 2022. ISSN 2227-7390. Disponível em: https://www.mdpi.com/2227-7390/10/10/1760.

CHAKRABORTY, U. K. **Computational intelligence in flow shop and job shop scheduling**. Missouri, USA: Springer, 2009. v. 230. ISBN 978-3-642-02835-9.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms**. 4. ed. Massachusetts, USA: MIT press, 2022.

DOANE, D. P.; SEWARD, L. W. **Applied statistics in business and economics**. 5. ed. New York, USA: Mcgraw-Hill, 2016.

DUDEK, R. A.; PANWALKAR, S.; SMITH, M. The lessons of flowshop scheduling research. **Operations Research**, v. 40, n. 1, p. 7–13, 1992.

EMMONS, H.; VAIRAKTARAKIS, G. **Flow shop scheduling: theoretical results, algorithms, and applications**. New York, USA: Springer Science & Business Media, 2013. v. 182. ISBN 978-1-4614-5151-8.

EPE. **Anuário Estatístico de Energia Elétrica**. 2024. Disponível em: https://www.epe.gov.br/pt/publicacoes-dados-abertos/publicacoes/anuario-estatistico-de-energia-eletrica. Acesso: 19/09/2024.

FEO, T. A.; RESENDE, M. G. Greedy randomized adaptive search procedures. **Journal of global optimization**, Springer, v. 6, p. 109–133, 1995.

FISHMAN, G. **Monte Carlo: concepts, algorithms, and applications**. New York, USA: Springer Science & Business Media, 1996. ISBN 0-387-94527.

FUCHIGAMI, H. Sequenciamento da produção em sistemas flow shop. **UFG, Goiânia-GO**, 2015.

GIMELFARB, M.; SANNER, S.; LEE, C.-G. ε **- BMC: A Bayesian Ensemble Approach to Epsilon-Greedy Exploration in Model-Free Reinforcement Learning**. 2020. Disponível em: https://arxiv.org/abs/2007.00869. Acesso em: 10 de outubro de 2024.

GLOVER, F. Tabu search—part i. **ORSA Journal on computing**, Informs, v. 1, n. 3, p. 190–206, 1989.

GLOVER, F. W.; KOCHENBERGER, G. A. **Handbook of metaheuristics**. [*S. l.*]: Springer Science & Business Media, 2003. v. 57.

GOLDBERG, D. E. Genetic algorithms in search. **Optimization, Machine Learning**, Addison-Wesley, 1989.

GONZALEZ-NEIRA, E. M.; FERONE, D.; HATAMI, S.; JUAN, A. A. A biased-randomized simheuristic for the distributed assembly permutation flowshop problem with stochastic processing times. **Simulation Modelling Practice and Theory**, v. 79, p. 23–36, 2017. ISSN 1569-190X. Disponível em: https://www.sciencedirect.com/science/article/pii/S1569190X17301338.

GONZALEZ-NEIRA, E. M.; MONTOYA-TORRES, J. R.; JIMENEZ, J.-F. A multicriteria simheuristic approach for solving a stochastic permutation flow shop scheduling problem. **Algorithms**, v. 14, n. 7, 2021. ISSN 1999-4893. Disponível em: https://www.mdpi.com/1999-4893/14/7/210.

GOURGAND, M.; GRANGEON, N.; NORRE, S. A contribution to the stochastic flow shop scheduling problem. **European Journal of Operational Research**, v. 151, n. 2, p. 415–433, 2003. ISSN 0377-2217. Meta-heuristics in combinatorial optimization. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221702008354.

GUO, D.; LIU, S.; LING, S.; LI, M.; JIANG, Y.; LI, M.; HUANG, G. Q. The marriage of operations research and reinforcement learning: Integration of neh into q-learning algorithm for the permutation flowshop scheduling problem. **Expert Systems with Applications**, v. 255, p. 124779, 2024. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417424016464.

GUZMAN, E.; ANDRES, B.; POLER, R. Models and algorithms for production planning, scheduling and sequencing problems: A holistic framework and a systematic review. **Journal of Industrial Information Integration**, v. 27, p. 100287, 2022. ISSN 2452-414X. Disponível em: https://www.sciencedirect.com/science/article/pii/S2452414X21000844.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H.; FRIEDMAN, J. H. **The elements of statistical learning: data mining, inference, and prediction**. 2. ed. [*S. l.*]: Springer, 2009.

HE, Z.; WANG, K.; LI, H.; SONG, H.; LIN, Z.; GAO, K.; SADOLLAH, A. Improved q-learning algorithm for solving permutation flow shop scheduling problems. **IET Collaborative Intelligent Manufacturing**, Wiley Online Library, v. 4, n. 1, p. 35–44, 2022.

HORDONES, P. A.; FUCHIGAMI, H. Programação da produção em flow shop permutacional envolvendo medidas de atraso: uma contribuição bibliométrica a partir da base de dados web of science. **HOLOS**, v. 7, p. 81–97, dez. 2017. Disponível em: https://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/5711.

IEA, I. E. A. **Electricity 2024: Analysis and Forecast to 2026**. 2024. Acesso em: 24 set. 2024. Disponível em: https://www.iea.org/reports/electricity-2024.

IGNALL, E.; SCHRAGE, L. Application of the branch and bound technique to some flow-shop scheduling problems. **Operations research**, INFORMS, v. 13, n. 3, p. 400–412, 1965.

ISTOKOVIC, D.; PERINIC, M.; VLATKOVIC, M.; BREZOCNIK, M. Minimizing total production cost in a hybrid flow shop: a simulation-optimization approach. **International journal of simulation modelling**, v. 19, n. 4, p. 559–570, 2020.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. **Electronic Markets**, Springer, v. 31, n. 3, p. 685–695, 2021.

JARADAT, G.; AYOB, M.; ALMARASHDEH, I. The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems. **Applied Soft Computing**, v. 44, p. 45–56, 2016. ISSN 1568-4946. Disponível em: https://www.sciencedirect.com/science/article/pii/S1568494616000041.

JOHNSON, S. M. Optimal two- and three-stage production schedules with setup times included. **Naval Research Logistics Quarterly**, v. 1, n. 1, p. 61–68, 1954. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800010110.

JUAN, A. A.; BARRIOS, B. B.; VALLADA, E.; RIERA, D.; JORBA, J. A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. **Simulation Modelling Practice and Theory**, Elsevier, v. 46, p. 101–117, 2014.

KARIMI-MAMAGHAN, M.; MOHAMMADI, M.; PASDELOUP, B.; MEYER, P. Learning to select operators in meta-heuristics: An integration of q-learning into the iterated greedy algorithm for the permutation flowshop scheduling problem. **European Journal of Operational Research**, v. 304, n. 3, p. 1296–1330, 2023. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221722002788.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671–680, 1983. Disponível em: https://www.science.org/doi/abs/10.1126/science.220.4598.671.

LAROSE, D. T.; LAROSE, C. D. **Discovering knowledge in data: an introduction to data mining**. 2. ed. New Jersey, USA: John Wiley & Sons, 2014. ISBN 978-0-470-90874-7.

LI, H.; GAO, K.; DUAN, P.-Y.; LI, J.-Q.; ZHANG, L. An improved artificial bee colony algorithm with q-learning for solving permutation flow-shop scheduling problems. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 53, n. 5, p. 2684–2693, 2023.

LI, X.; LI, M. Multiobjective local search algorithm-based decomposition for multiobjective permutation flow shop scheduling problem. **IEEE Transactions on Engineering Management**, IEEE, v. 62, n. 4, p. 544–557, 2015.

LIANG, Z.; ZHONG, P.; LIU, M.; ZHANG, C.; ZHANG, Z. A computational efficient optimization of flow shop scheduling problems. **Scientific Reports**, Nature Publishing Group UK London, v. 12, n. 1, p. 845, 2022.

LIN, S. Computer solutions of the traveling salesman problem. **The Bell System Technical Journal**, v. 44, n. 10, p. 2245–2269, 1965.

LIU, G.-S.; LI, J.-J.; TANG, Y.-S. Minimizing total idle energy consumption in the permutation flow shop scheduling problem. **Asia-Pacific Journal of Operational Research**, v. 35, n. 06, p. 1850041, 2018. Disponível em: https://doi.org/10.1142/S0217595918500410.

LIU, L.; URGO, M. A branch-and-bound approach to minimise the value-at-risk of the makespan in a stochastic two-machine flow shop. **International Journal of Production Research**, Taylor & Francis, v. 62, n. 6, p. 2107–2123, 2024.

LU, Y.; YUAN, Y.; SITAHONG, A.; CHAO, Y.; WANG, Y. An optimization method for green permutation flow shop scheduling based on deep reinforcement learning and moea/d. **Machines**, v. 12, n. 10, 2024. ISSN 2075-1702. Disponível em: https://www.mdpi.com/2075-1702/12/10/721.

LóPEZ, O. A. M.; LóPEZ, A. M.; CROSSA, J. **Multivariate statistical machine learning methods for genomic prediction**. Colima, México: Springer Nature, 2022. ISBN 978-3-030-89009-4.

MACCARRONE, L.; GIOVANNELLI, T.; FERONE, D.; PANADERO, J.; JUAN, A. A. A simheuristic algorithm for solving an integrated resource allocation and scheduling problem. In: **2018 Winter Simulation Conference (WSC)**. [*S. l.*: *s. n.*], 2018. p. 3340–3351.

MARCHESANO, M. G.; GUIZZI, G.; SANTILLO, L. C.; VESPOLI, S. Dynamic scheduling in a flow shop using deep reinforcement learning. In: DOLGUI, A.; BERNARD, A.; LEMOINE, D.; VONCIEMINSKI, G.; ROMERO, D. (Ed.). **ADVANCES IN PRODUCTION MANAGEMENT**

SYSTEMS: ARTIFICIAL INTELLIGENCE FOR SUSTAINABLE AND RESILIENT PRODUCTION SYSTEMS, APMS 2021, PT I. [*S. l.*], 2021. (IFIP Advances in Information and Communication Technology, v. 630), p. 152–160. ISBN 978-3-030-85874-2; 978-3-030-85873-5. ISSN 1868-4238. International-Federation-of-Information-Processing-Working-Group-5.7 (IFIP WG 5.7) International Conference on Advances in Production Management Systems (APMS), ELECTR NETWORK, SEP 05-09, 2021.

MARTIN, P. **Linear regression: An introduction to statistical models**. 1. ed. London, UK: SAGE Publications Ltd, 2021. ISBN 978-1-5264-2417-4.

METROPOLIS, N.; ULAM, S. The monte carlo method. **Journal of the American statistical association**, Taylor & Francis, v. 44, n. 247, p. 335–341, 1949.

MICHIE, D.; SPIEGELHALTER, D. J.; TAYLOR, C. C.; CAMPBELL, J. **Machine learning, neural and statistical classification**. [*S. l.*]: Ellis Horwood, 1995.

MLADENOVIć, N.; HANSEN, P. Variable neighborhood search. **Computers & Operations Research**, v. 24, n. 11, p. 1097–1100, 1997. ISSN 0305-0548. Disponível em: https://www.sciencedirect.com/science/article/pii/S0305054897000312.

MRABET, M. A. E.; MAKKAOUI, K. E.; FAIZE, A. Supervised machine learning: A survey. In: **2021 4th International Conference on Advanced Communication Technologies and Networking (CommNet)**. [*S. l.*: *s. n.*], 2021. p. 1–10.

MURATA, T.; ISHIBUCHI, H.; TANAKA, H. Genetic algorithms for flowshop scheduling problems. **Computers & Industrial Engineering**, v. 30, n. 4, p. 1061–1071, 1996. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/0360835296000538.

MüLLER, A.; GRUMBACH, F.; KATTENSTROTH, F. Reinforcement learning for two-stage permutation flow shop scheduling—a real-world application in household appliance production. **IEEE Access**, v. 12, p. 11388–11399, 2024.

NAGARAJAN, V.; SVIRIDENKO, M. Tight bounds for permutation flow shop scheduling. In: LODI, A.; PANCONESI, A.; RINALDI, G. (Ed.). **Integer Programming and Combinatorial Optimization**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 154–168. ISBN 978-3-540-68891-4.

NAWAZ, M.; ENSCORE, E. E.; HAM, I. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. **Omega**, v. 11, n. 1, p. 91–95, 1983. ISSN 0305-0483. Disponível em: https://www.sciencedirect.com/science/article/pii/0305048383900889.

NOWICKI, E.; SMUTNICKI, C. A fast tabu search algorithm for the permutation flow-shop problem. **European Journal of Operational Research**, v. 91, n. 1, p. 160–175, 1996. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/0377221795000372.

OLUYISOLA, O. E.; BHALLA, S.; SGARBOSSA, F.; STRANDHAGEN, J. O. Designing and developing smart production planning and control systems in the industry 4.0 era: a methodology and case study. **Journal of Intelligent Manufacturing**, Springer, v. 33, n. 1, p. 311–332, 2022.

OSMAN, I.; POTTS, C. Simulated annealing for permutation flow-shop scheduling. **Omega**, v. 17, n. 6, p. 551–557, 1989. ISSN 0305-0483. Disponível em: https://www.sciencedirect.com/science/article/pii/0305048389900595.

OSMAN, I. H.; KELLY, J. P. Meta-heuristics: an overview. **Meta-heuristics: Theory and applications**, Springer, p. 1–21, 1996.

ÖZTOP, H.; TASGETIREN, M. F.; ELIYIĞI, D. T.; PAN, Q.-K.; KANDILLER, L. An energy-efficient permutation flowshop scheduling problem. **Expert Systems with Applications**, v. 150, p. 113279, 2020. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420301044.

PALMER, D. S. Sequencing jobs through a multi-stage process in the minimum total time—a quick method of obtaining a near optimum. **Journal of the Operational Research Society**, Taylor & Francis, v. 16, n. 1, p. 101–107, 1965. Disponível em: https://doi.org/10.1057/jors.1965.8.

PAN, Z.; WANG, L.; WANG, J.; LU, J. Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. **IEEE Transactions on Emerging Topics in Computational Intelligence**, v. 7, n. 4, p. 983–994, 2023.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 5. ed. New York, USA: Springer Science & Business Media, 2016. ISBN 978-3-319-26578-0.

POTTS, C. An adaptive branching rule for the permutation flow-shop problem. **European Journal of Operational Research**, v. 5, n. 1, p. 19–25, 1980. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/0377221780900697.

QIU, R.; SUN, Z.; YANG, Y. Dimes: A differentiable meta solver for combinatorial optimization problems. In: KOYEJO, S.; MOHAMED, S.; AGARWAL, A.; BELGRAVE, D.; CHO, K.; OH, A. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2022. v. 35, p. 25531–25546. Disponível em: https://proceedings.neurips.cc/paper_files/paper/2022/file/a3a7387e49f4de290c23beea2dfcdc75-Paper-Conference.pdf.

RAMANAN, T. R.; IQBAL, M.; UMARALI, K. A particle swarm optimization approach for permutation flow shop scheduling problem. **International Journal for Simulation and Multidisciplinary Design Optimization**, EDP Sciences, v. 5, p. A20, 2014.

RAVETTI, M. G.; RIVEROS, C.; MENDES, A.; RESENDE, M. G.; PARDALOS, P. M. Parallel hybrid heuristics for the permutation flow shop problem. **Annals of Operations Research**, Springer, v. 199, p. 269–284, 2012.

RAYCHAUDHURI, S. Introduction to monte carlo simulation. In: **2008 Winter Simulation Conference**. [*S. l.*: *s. n.*], 2008. p. 91–100.

REN, J. F.; YE, C. M.; LI, Y. A new solution to distributed permutation flow shop scheduling problem based on nash q-learning. **ADVANCES IN PRODUCTION ENGINEERING & MANAGEMENT**, v. 16, n. 3, p. 269–284, SEP 2021. ISSN 1854-6250.

RODRIGUEZ-ESPINOSA, D. F.; CRUZ-VARGAS, D.; DELGADO-MERCHAN, D. E.; GONZALEZ-ESTUPINAN, D. H.; GONZALEZ-NEIRA, E. M. Nsga-ii simheuristic to solve a multi-objective flexible flow shop problem under stochastic machine breakdowns. **JOURNAL OF PROJECT MANAGEMENT**, v. 9, n. 4, p. 493–512, 2024. ISSN 2371-8366.

ROMLI, F.; HARMIN, M. Y. Use of monte carlo method to estimate subsystem redesign risk for complex products: aircraft redesign case study. **Aircraft Engineering and Aerospace Technology: An International Journal**, Emerald Group Publishing Limited, v. 87, n. 6, p. 563–570, 2015.

RUBINSTEIN, R. Y.; KROESE, D. P. **Simulation and the Monte Carlo method**. 3. ed. New Jersey, USA: John Wiley & Sons, 2017. ISBN 9781118632208.

RíOS-MERCADO, R. Z.; BARD, J. F. Heuristics for the flow line problem with setup costs. **European Journal of Operational Research**, v. 110, n. 1, p. 76–98, 1998. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221797002130.

SCHALLER, J.; VALENTE, J. M. Heuristics for scheduling jobs in a permutation flow shop to minimize total earliness and tardiness with unforced idle time allowed. **Expert Systems with Applications**, v. 119, p. 376–386, 2019. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417418307279.

SHAFIQ, W. Optimizing organizational performance: A data-driven approach in management science. **Bulletin of Management Review**, v. 1, n. 2, p. 31–40, Jun. 2024. Disponível em: https://bulletinofmanagement.com/index.php/Journal/article/view/48.

SIGAUD, O.; BUFFET, O. **Markov decision processes in artificial intelligence**. London, UK: John Wiley & Sons, 2010.

SÖRENSEN, K.; GLOVER, F. Metaheuristics. **Encyclopedia of operations research and management science**, Springer Boston, MA, USA, v. 62, p. 960–970, 2013.

SOUZA, B. L.; MOTA, M. R. R.; KRAMER, R. H. F. H. R.; LINS, I. D. Aplicação de métodos de regressão associados a uma meta-heurística no contexto do flowshop permutacional. **Anais do Encontro Nacional de Engenharia de Produção - Enegep**, 2024.

STÜTZLE, T. **Applying iterated local search to the permutation flow shop problem**. [*S. l.*], 1998.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. Massachusetts, USA: MIT Press, 2018. ISBN 9780262039246.

TAILLARD, E. Some efficient heuristic methods for the flow shop sequencing problem. **European Journal of Operational Research**, v. 47, n. 1, p. 65–74, 1990. ISSN 0377-2217. Disponível em: https://www.sciencedirect.com/science/article/pii/037722179090090X.

TAILLARD, E. **Scheduling instances: benchmarks for basic scheduling problems**. 1993. Acessado em: 20 jan. 2025. Disponível em: http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html.

TALBI, E.-G. Metaheuristics: From design to implementation. **John Wiley & Sons google schola**, v. 2, p. 268–308, 2009.

UHLIG, S.; ALKHASLI, I.; SCHUBERT, F.; TSCHÖPE, C.; WOLFF, M. A review of synthetic and augmented training data for machine learning in ultrasonic non-destructive evaluation. **Ultrasonics**, Elsevier, v. 134, p. 107041, 2023.

USLU, C. A.; DENGIZ, B.; AGLAN, C.; SABUNCUOGLU, I. Modified self-adaptive local search algorithm for a biobjective permutation flow shop scheduling problem. **TURKISH JOURNAL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES**, v. 27, n. 4, p. 2730–2745, 2019. ISSN 1300-0632.

VELDE, S. van de. Minimizing the sum of the job completion times in the two-machine flow shop by lagrangian relaxation. **Annals of Operations Research**, v. 26, p. 257–268, 1990. ISSN 1572-9338. Disponível em: https://doi.org/10.1007/BF03500931.

VILLARINHO, P. A.; PANADERO, J.; PESSOA, L. S.; JUAN, A. A.; OLIVEIRA, F. L. C. A simheuristic algorithm for the stochastic permutation flow-shop problem with delivery dates and cumulative payoffs. **International Transactions in Operational Research**, Wiley Online Library, v. 28, n. 2, p. 716–737, 2021.

WATKINS, C. J.; DAYAN, P. Q-learning. **Machine learning**, Springer, v. 8, p. 279–292, 1992.

WATKINS, C. J. C. H. **Learning from delayed rewards**. [*S. l.*]: King's College, Cambridge United Kingdom, 1989.

XIAO, Y.-Y.; ZHANG, R.-Q.; ZHAO, Q.-H.; KAKU, I. Permutation flow shop scheduling with order acceptance and weighted tardiness. **Applied Mathematics and Computation**, v. 218, n. 15, p. 7911–7926, 2012. ISSN 0096-3003. Disponível em: https://www.sciencedirect.com/science/article/pii/S0096300312001245.

YAMASHIRO, H.; NONAKA, H. Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem. **Operations Research Perspectives**, v. 8, p. 100196, 2021. ISSN 2214-7160. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214716021000178.

YAN, X.; SU, X. **Linear regression analysis: theory and computing**. Hackensack, USA: world scientific, 2009. ISBN 13: 978-981-283-410-2.

YING, K.-C.; POURHEJAZY, P.; CHENG, S.-H. Reinforcement learning-based alpha-list iterated greedy for production scheduling. **Intelligent Systems with Applications**, v. 24, p. 200451, 2024. ISSN 2667-3053. Disponível em: https://www.sciencedirect.com/science/article/pii/S266730532400125X.

ZAIED, A. N. H.; ISMAIL, M. M.; MOHAMED, S. S. Permutation flow shop scheduling problem with makespan criterion: literature review. **J. Theor. Appl. Inf. Technol**, v. 99, n. 4, p. 830–848, 2021.

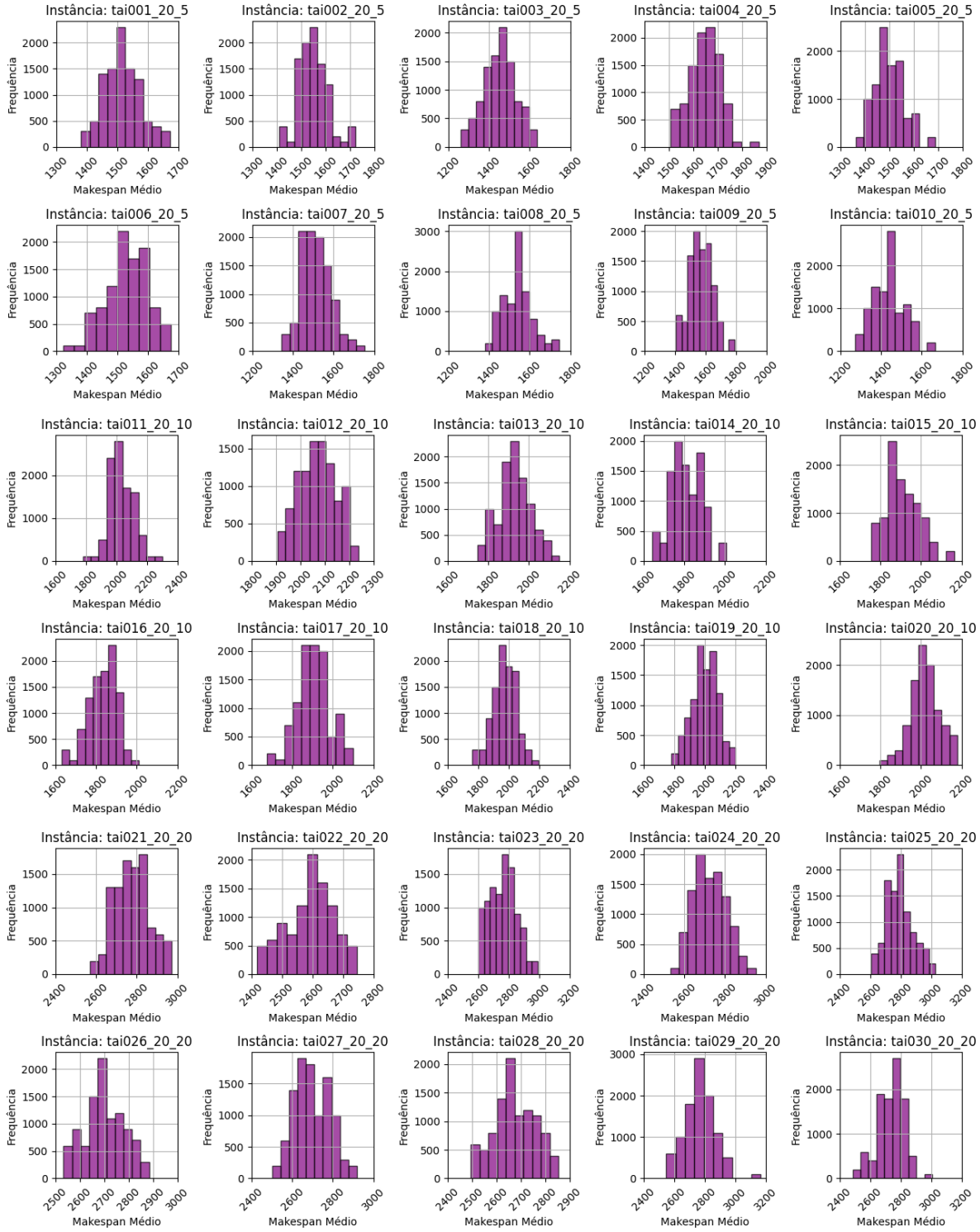
ZHONG, R. Y.; NEWMAN, S. T.; HUANG, G. Q.; LAN, S. Big data for supply chain management in the service and manufacturing sectors: Challenges, opportunities, and future perspectives. **Computers & Industrial Engineering**, v. 101, p. 572–591, 2016. ISSN 0360-8352. Disponível em: https://www.sciencedirect.com/science/article/pii/S0360835216302388.

ZHOU, T.; LUO, L.; JI, S.; HE, Y. A reinforcement learning approach to robust scheduling of permutation flow shop. **Biomimetics**, v. 8, n. 6, 2023. ISSN 2313-7673. Disponível em: https://www.mdpi.com/2313-7673/8/6/478.

APÊNDICE A – CARACTERIZAÇÃO DOS DADOS DE TREINAMENTO

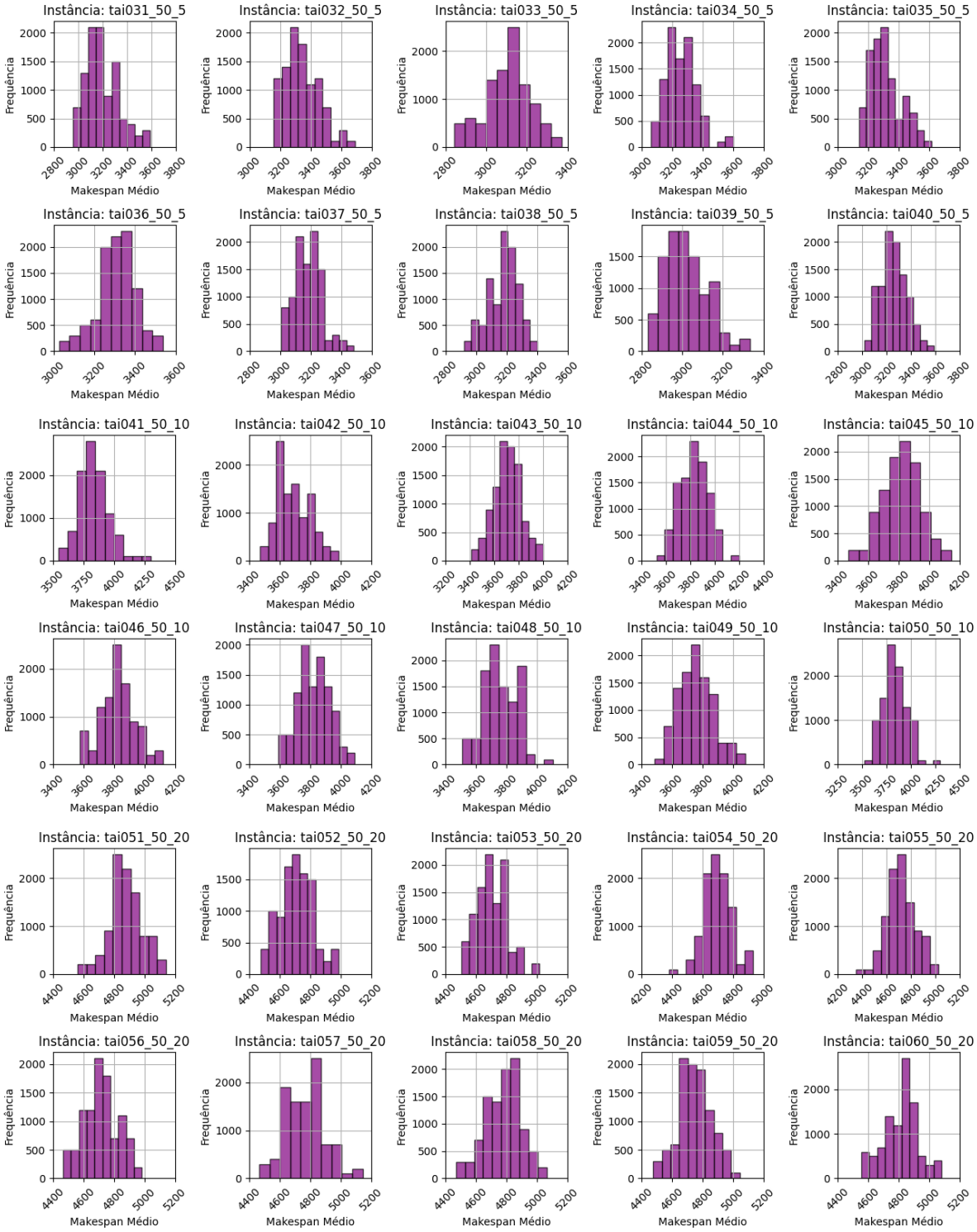
Neste apêndice, encontram-se as informações pertinentes aos dados utilizados para treinamento do modelo de RLM, operacionalizaram os dois modelos híbridos desenvolvidos nesta dissertação: MSNEHT-RLM-POOL e GRASP-RLM-POOL.

Figura 34 – Histograma das instânicas tai001 a tai030



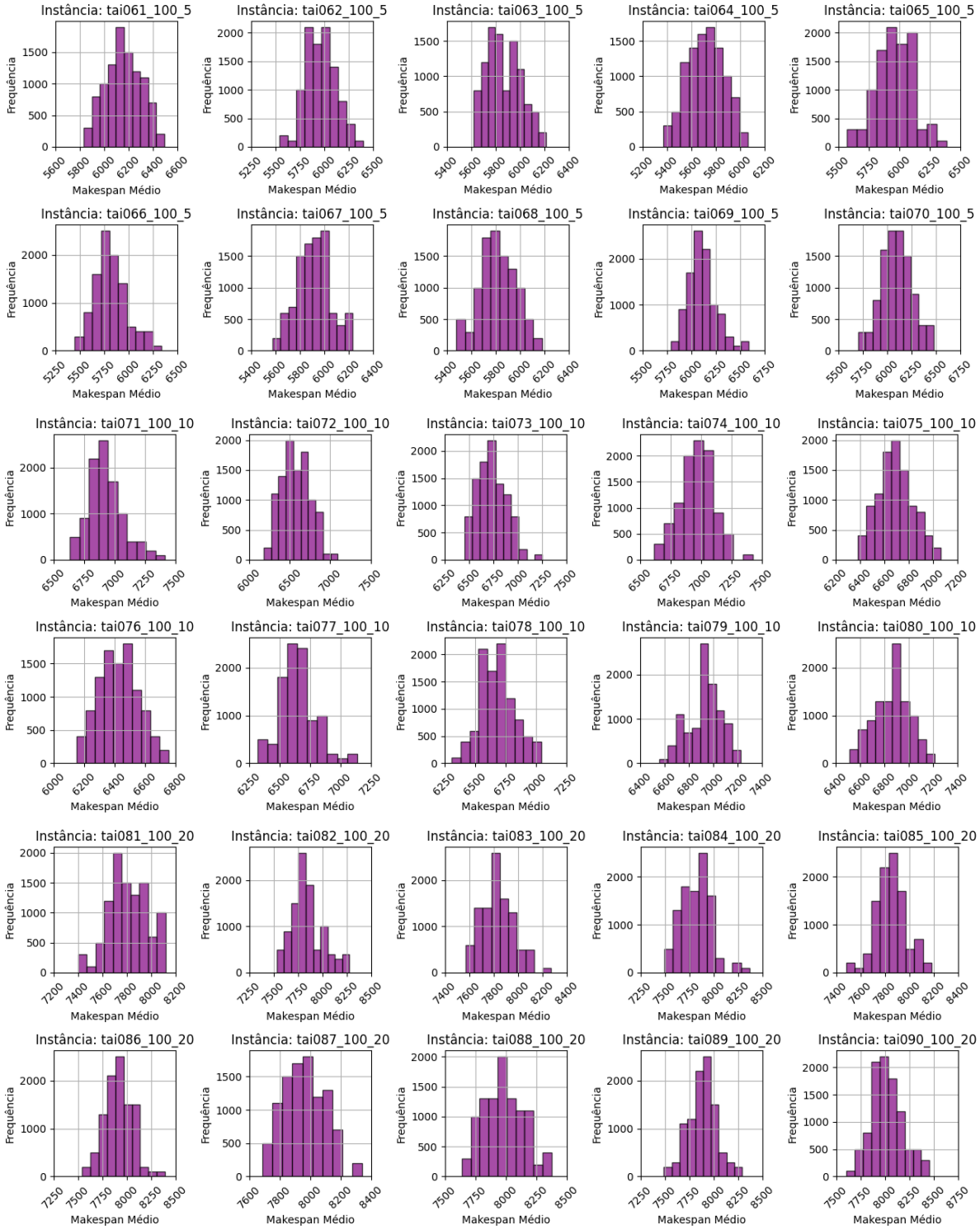
Fonte: Elaborada pelo autor (2025)

Figura 35 – Histograma das instânicas tai031 a tai060



Fonte: Elaborada pelo autor (2025)

Figura 36 – Histograma das instânicas tai061 a tai090



Fonte: Elaborada pelo autor (2025)

Tabela 13 – Estatística descritiva referente aos dados de treinamento de cada instância

Instância	Média	Desvio-padrão	Mínimo	Máximo	25%	50%	75%	Qtd
tai001_20_5	1516,59	59,02	1380,59	1672,76	1471,25	1512,85	1555,79	10000

Continua na página seguinte

Tabela 13 – Estatística Descritiva Referente aos Dados de Treinamento de Cada Instância

(Continuação)

Instância	Média	Desvio-padrão	Mínimo	Máximo	25%	50%	75%	Qtd
tai002_20_5	1548,61	59,60	1409,20	1722,39	1506,22	1548,45	1582,85	10000
tai003_20_5	1451,61	79,50	1259,92	1634,60	1392,54	1453,78	1505,59	10000
tai004_20_5	1647,80	63,38	1507,22	1869,14	1611,88	1648,96	1693,73	10000
tai005_20_5	1498,83	61,98	1361,77	1688,89	1460,36	1492,63	1538,51	10000
tai006_20_5	1533,20	68,10	1323,31	1676,74	1486,49	1532,68	1585,71	10000
tai007_20_5	1512,40	72,52	1344,58	1752,31	1461,41	1507,07	1559,96	10000
tai008_20_5	1540,13	72,57	1379,35	1745,16	1488,40	1540,92	1570,11	10000
tai009_20_5	1568,14	75,36	1405,95	1792,28	1518,64	1567,13	1618,69	10000
tai010_20_5	1439,10	79,11	1272,10	1665,77	1385,34	1439,95	1477,52	10000
tai011_20_10	2031,12	79,75	1778,27	2303,24	1982,47	2024,42	2087,18	10000
tai012_20_10	2068,09	76,25	1905,20	2237,07	2009,24	2070,13	2119,59	10000
tai013_20_10	1930,17	81,40	1745,53	2149,99	1882,02	1930,09	1982,15	10000
tai014_20_10	1809,71	75,65	1641,77	2005,45	1751,88	1810,40	1874,41	10000
tai015_20_10	1910,08	82,54	1755,48	2162,83	1859,34	1890,53	1968,87	10000
tai016_20_10	1831,36	70,75	1631,62	2010,27	1787,12	1840,98	1882,60	10000
tai017_20_10	1905,62	79,33	1675,80	2098,37	1850,98	1898,11	1956,50	10000
tai018_20_10	1972,56	77,91	1762,83	2194,33	1921,07	1973,79	2025,87	10000
tai019_20_10	1999,38	81,91	1782,45	2200,72	1946,02	1995,92	2052,77	10000
tai020_20_10	2019,55	72,42	1794,45	2178,04	1978,83	2022,43	2064,38	10000
tai021_20_20	2777,52	86,82	2569,53	2971,59	2709,07	2773,85	2836,96	10000
tai022_20_20	2591,74	76,05	2417,25	2746,49	2540,65	2602,32	2643,36	10000
tai023_20_20	2762,72	88,23	2599,43	2993,05	2689,19	2761,31	2821,54	10000
tai024_20_20	2724,28	81,05	2530,93	2950,85	2660,41	2721,51	2781,67	10000
tai025_20_20	2791,68	89,69	2607,70	3027,85	2722,81	2784,45	2844,65	10000
tai026_20_20	2700,31	82,00	2532,84	2884,16	2648,15	2696,45	2751,82	10000
tai027_20_20	2696,96	88,17	2501,65	2917,73	2637,27	2681,96	2770,89	10000
tai028_20_20	2676,25	84,53	2494,39	2855,20	2625,44	2671,60	2744,49	10000

Continua na página seguinte

Tabela 13 – Estatística Descritiva Referente aos Dados de Treinamento de Cada Instância

(Continuação)

Instância	Média	Desvio-padrão	Mínimo	Máximo	25%	50%	75%	Qtd
tai029_20_20	2771,90	98,25	2546,61	3166,50	2715,35	2763,38	2831,51	10000
tai030_20_20	2736,01	88,67	2483,23	3006,84	2682,97	2745,66	2795,29	10000
tai031_50_5	3199,27	134,74	2957,05	3590,06	3106,67	3175,67	3283,28	10000
tai032_50_5	3341,21	111,55	3157,75	3692,44	3261,54	3327,00	3398,69	10000
tai033_50_5	3105,28	111,04	2841,31	3371,36	3042,60	3123,78	3172,88	10000
tai034_50_5	3257,99	98,82	3067,89	3462,88	3178,86	3261,96	3330,29	10000
tai035_50_5	3195,03	133,71	2875,49	3505,88	3087,00	3185,11	3297,88	10000
tai036_50_5	3318,73	114,21	3091,13	3506,89	3228,06	3308,74	3362,74	10000
tai037_50_5	3262,59	134,36	2964,74	3537,77	3163,36	3242,45	3353,79	10000
tai038_50_5	3149,48	129,82	2848,09	3476,04	3061,96	3148,19	3231,93	10000
tai039_50_5	3172,90	141,78	2927,54	3538,53	3074,90	3161,82	3245,70	10000
tai040_50_5	3286,07	126,43	3019,53	3532,01	3181,59	3270,63	3330,36	10000
tai041_50_10	4065,90	145,52	3722,48	4482,93	3976,15	4067,23	4156,90	10000
tai042_50_10	4005,81	140,81	3733,08	4380,95	3906,32	3997,59	4076,90	10000
tai043_50_10	4048,86	143,61	3723,53	4387,01	3956,21	4046,83	4142,34	10000
tai044_50_10	3917,07	135,63	3711,76	4270,55	3824,43	3916,65	3996,10	10000
tai045_50_10	3976,68	139,37	3730,35	4373,45	3886,29	3969,69	4056,02	10000
tai046_50_10	3989,75	151,07	3665,85	4330,02	3881,85	3968,52	4064,45	10000
tai047_50_10	3972,37	149,02	3735,29	4307,75	3868,51	3963,90	4056,50	10000
tai048_50_10	4066,43	146,49	3716,53	4386,93	3973,55	4056,69	4151,66	10000
tai049_50_10	4046,77	141,69	3740,93	4372,75	3951,75	4038,58	4124,57	10000
tai050_50_10	4027,65	140,15	3764,00	4401,85	3947,64	4023,75	4093,99	10000
tai051_50_20	5485,75	156,12	5178,19	5822,35	5391,48	5483,57	5576,77	10000
tai052_50_20	5415,36	156,06	5096,34	5722,16	5321,78	5413,85	5508,98	10000
tai053_50_20	5353,97	157,61	5036,09	5732,20	5247,15	5350,78	5446,74	10000
tai054_50_20	5380,69	157,91	5103,58	5711,14	5277,83	5379,85	5463,37	10000
tai055_50_20	5349,52	162,82	5062,85	5783,31	5251,45	5350,07	5451,92	10000

Continua na página seguinte

Tabela 13 – Estatística Descritiva Referente aos Dados de Treinamento de Cada Instância

(Continuação)

Instância	Média	Desvio-padrão	Mínimo	Máximo	25%	50%	75%	Qtd
tai056_50_20	5314,94	159,52	5023,82	5686,23	5221,18	5316,62	5401,84	10000
tai057_50_20	5468,82	155,77	5147,87	5824,06	5384,75	5468,02	5554,70	10000
tai058_50_20	5332,77	163,09	5074,30	5800,98	5234,78	5324,02	5431,56	10000
tai059_50_20	5331,68	162,52	5084,03	5790,92	5231,68	5324,96	5425,10	10000
tai060_50_20	5360,89	163,60	5118,59	5823,89	5265,61	5355,75	5446,71	10000
tai061_100_5	6159,08	146,58	5832,20	6495,15	6065,64	6157,68	6273,91	10000
tai062_100_5	5963,91	153,74	5538,81	6402,60	5859,42	5959,10	6065,46	10000
tai063_100_5	5867,07	142,99	5620,00	6211,80	5748,47	5842,36	5967,64	10000
tai064_100_5	5714,36	148,00	5371,18	6065,10	5619,83	5714,79	5814,11	10000
tai065_100_5	5964,04	149,43	5568,02	6390,66	5869,37	5965,95	6069,61	10000
tai066_100_5	5821,15	170,80	5450,28	6339,50	5693,68	5797,10	5903,41	10000
tai067_100_5	5914,38	142,29	5574,57	6234,67	5824,47	5909,65	5992,00	10000
tai068_100_5	5818,77	151,72	5478,32	6181,50	5710,49	5810,24	5952,66	10000
tai069_100_5	6112,30	147,19	5788,08	6589,48	6019,60	6095,36	6179,53	10000
tai070_100_5	6091,14	157,69	5700,89	6475,51	5979,90	6092,11	6200,94	10000
tai071_100_10	6931,02	144,95	6634,57	7416,11	6844,58	6907,96	7002,15	10000
tai072_100_10	6571,61	175,08	6178,01	7093,06	6446,91	6563,95	6695,03	10000
tai073_100_10	6736,17	151,26	6451,31	7254,35	6618,57	6724,24	6842,76	10000
tai074_100_10	6968,17	142,19	6611,56	7427,58	6873,53	6982,68	7054,78	10000
tai075_100_10	6693,32	145,73	6383,89	7066,62	6596,29	6683,56	6783,57	10000
tai076_100_10	6424,07	129,28	6150,93	6757,41	6334,20	6415,92	6511,72	10000
tai077_100_10	6646,95	148,07	6317,11	7138,38	6556,53	6630,73	6713,18	10000
tai078_100_10	6689,82	145,35	6306,74	7045,75	6584,78	6680,86	6767,88	10000
tai079_100_10	6928,73	134,07	6558,81	7224,86	6852,59	6931,79	7019,83	10000
tai080_100_10	6861,50	145,68	6513,06	7216,99	6760,60	6875,18	6960,39	10000
tai081_100_20	7808,77	155,86	7397,99	8118,36	7704,46	7799,72	7912,70	10000
tai082_100_20	7844,45	164,59	7530,79	8278,27	7737,11	7818,71	7923,51	10000

Continua na página seguinte

Tabela 13 – Estatística Descritiva Referente aos Dados de Treinamento de Cada Instância

(Continuação)

Instância	Média	Desvio-padrão	Mínimo	Máximo	25%	50%	75%	Qtd
tai083_100_20	7836,65	133,68	7567,98	8273,74	7755,72	7834,88	7919,02	10000
tai084_100_20	7813,09	153,06	7491,76	8369,06	7698,63	7813,92	7916,86	10000
tai085_100_20	7849,32	125,36	7479,38	8181,71	7772,86	7851,14	7911,74	10000
tai086_100_20	7919,01	147,39	7542,35	8393,86	7822,18	7916,11	8012,38	10000
tai087_100_20	7959,42	134,88	7687,11	8341,56	7867,19	7952,41	8039,33	10000
tai088_100_20	7980,01	161,69	7637,04	8374,91	7857,63	7976,35	8097,93	10000
tai089_100_20	7885,19	147,02	7489,27	8292,23	7798,55	7882,67	7970,48	10000
tai090_100_20	8018,69	162,01	7606,16	8457,53	7908,18	8000,92	8115,81	10000
tai091_200_10	12323,95	189,76	11892,32	12771,69	12203,35	12304,91	12439,32	10000
tai092_200_10	12263,32	200,48	11786,19	12699,08	12121,86	12240,98	12402,77	10000
tai093_200_10	12466,10	225,15	11960,15	13046,22	12324,53	12463,96	12590,99	10000
tai094_200_10	12291,67	205,13	11822,77	12904,65	12169,87	12276,42	12415,15	10000
tai095_200_10	12310,95	224,41	11775,25	12822,56	12138,03	12276,24	12466,45	10000
tai096_200_10	12024,92	218,04	11630,40	12585,23	11875,78	11989,21	12186,00	10000
tai097_200_10	12486,96	209,84	11903,39	12902,82	12355,61	12471,34	12648,84	10000
tai098_200_10	12465,00	183,74	12092,69	13094,56	12344,52	12464,80	12541,05	10000
tai099_200_10	12160,14	197,48	11661,37	12707,14	12026,76	12169,41	12290,66	10000
tai100_200_10	12350,19	194,76	11976,34	12908,70	12206,34	12334,20	12481,10	10000
tai101_200_20	13556,63	189,15	13069,56	14214,32	13439,59	13535,88	13639,66	10000
tai102_200_20	13788,79	194,88	13431,87	14429,22	13668,63	13772,30	13895,41	10000
tai103_200_20	13820,31	208,21	13337,75	14313,92	13675,26	13793,58	13954,69	10000
tai104_200_20	13784,44	227,60	13177,19	14274,26	13633,46	13763,52	13928,73	10000
tai105_200_20	13711,56	204,99	13168,59	14498,01	13572,35	13725,05	13816,05	10000
tai106_200_20	13765,96	206,43	13311,42	14202,42	13609,58	13757,52	13904,42	10000
tai107_200_20	13871,25	204,63	13429,37	14524,17	13736,88	13874,12	13984,63	10000
tai108_200_20	13756,93	202,99	13313,83	14418,81	13612,54	13733,18	13895,81	10000
tai109_200_20	13689,38	210,69	13239,49	14153,67	13541,94	13692,78	13850,92	10000

Continua na página seguinte

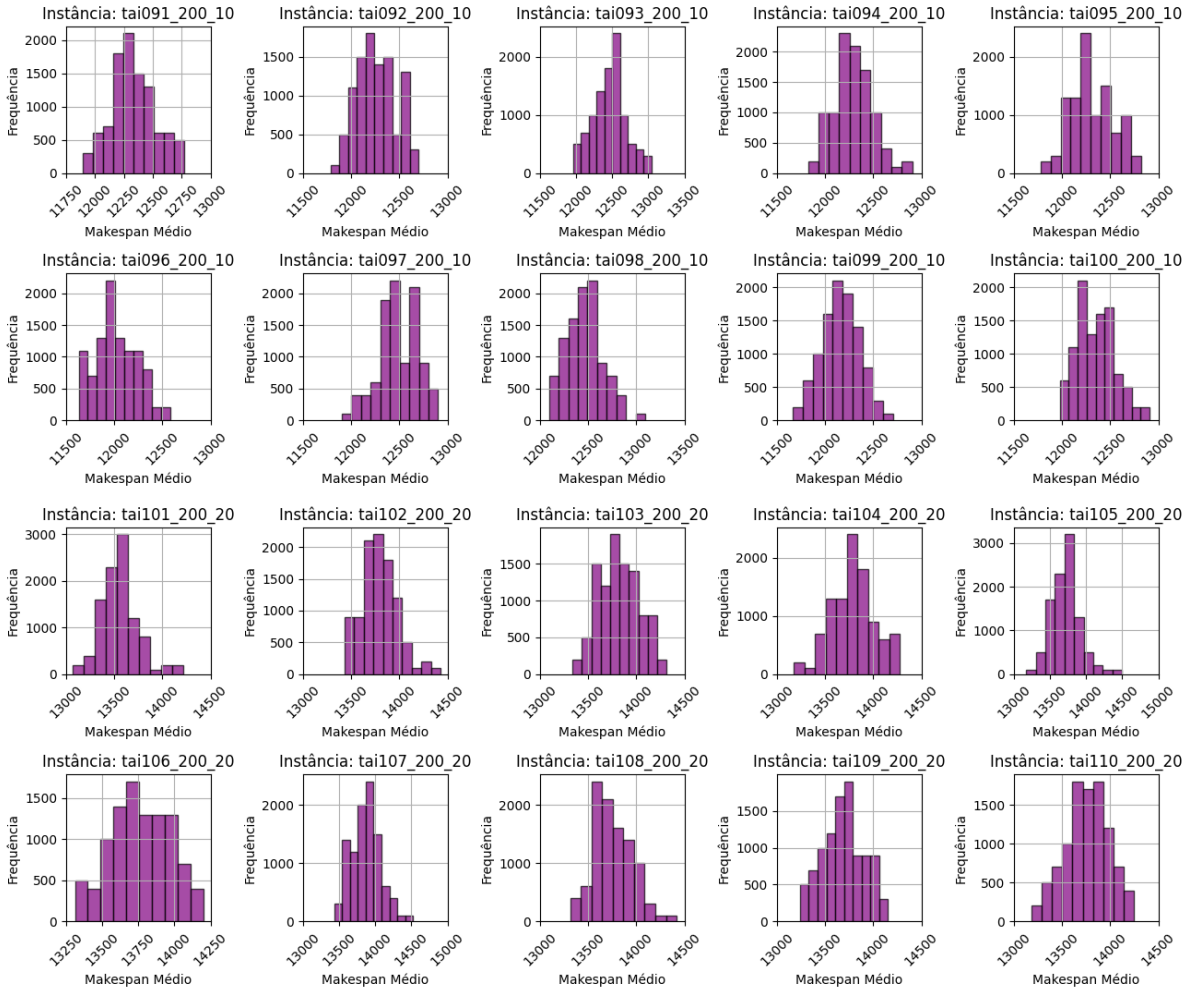
Tabela 13 – Estatística Descritiva Referente aos Dados de Treinamento de Cada Instância

(Continuação)

Instância	**Média**	**Desvio-padrão**	**Mínimo**	**Máximo**	**25%**	**50%**	**75%**	**Qtd**
tai110_200_20	13752,27	227,83	13183,90	14245,79	13613,17	13744,94	13912,47	10000

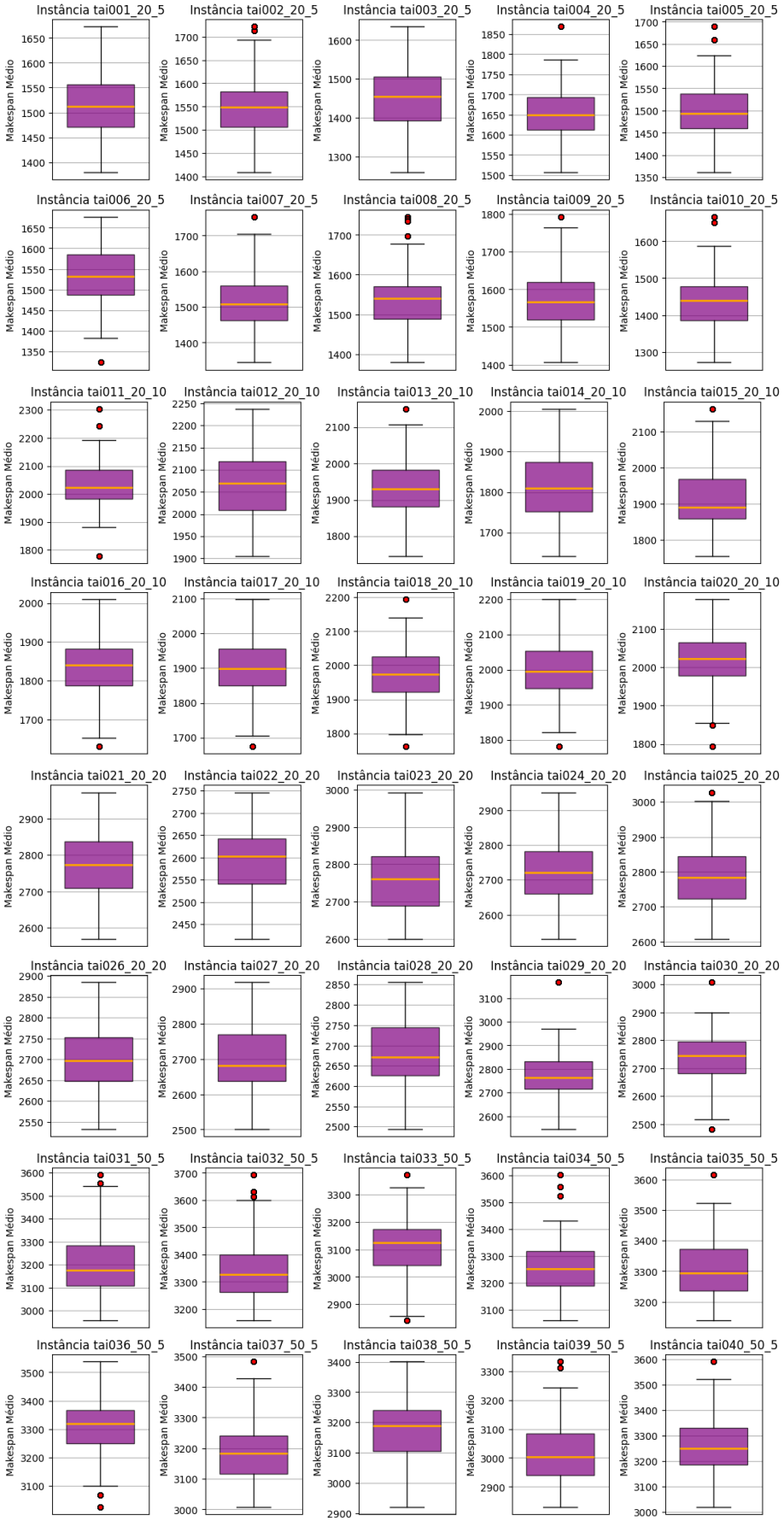
Fonte: Elaborada pelo autor (2025)

Figura 37 – Histograma das instânicas tai091 a tai110



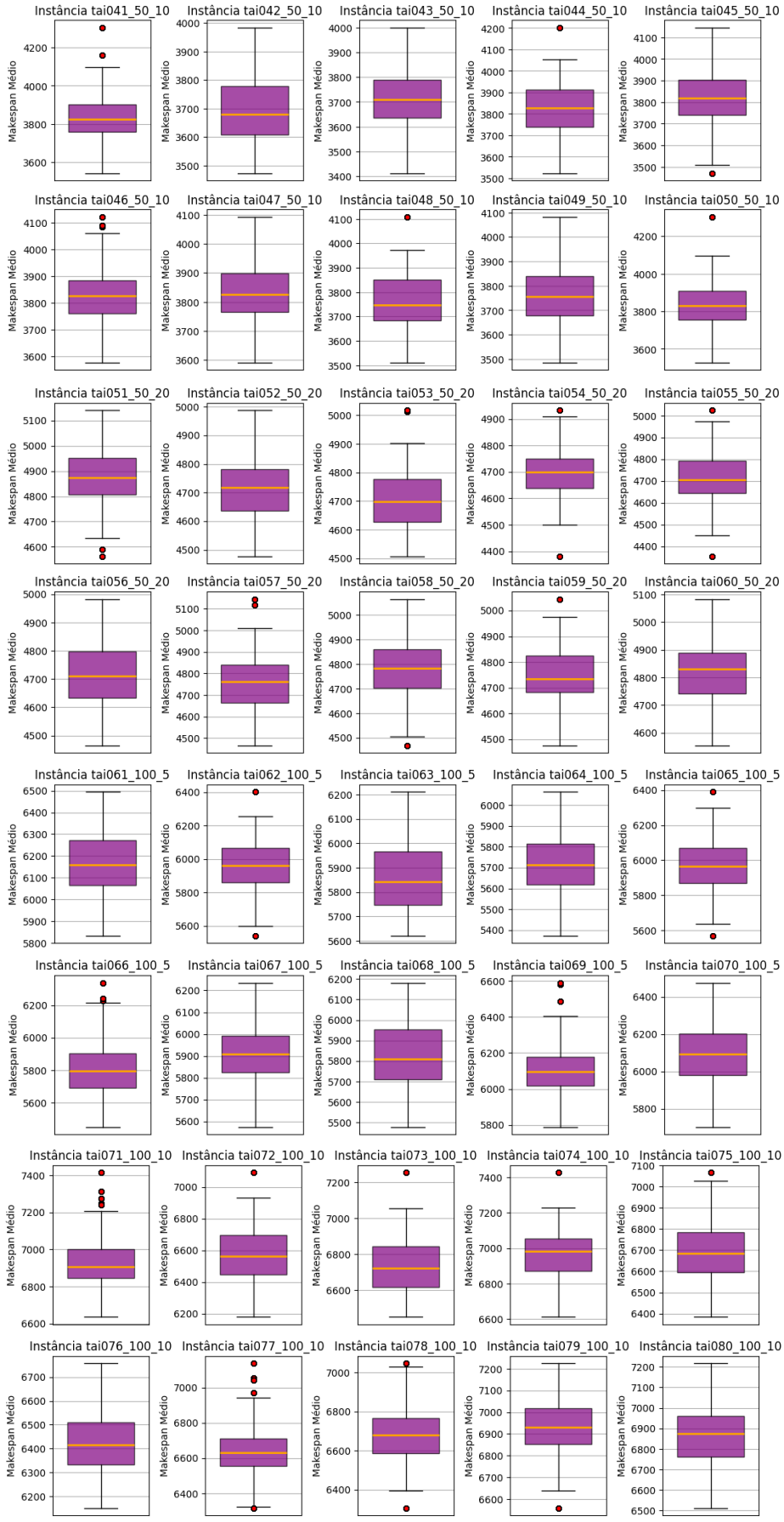
Fonte: Elaborada pelo autor (2025)

Figura 38 – Box-plot dos dados de treinamento das instâncias de tai001 a tai040



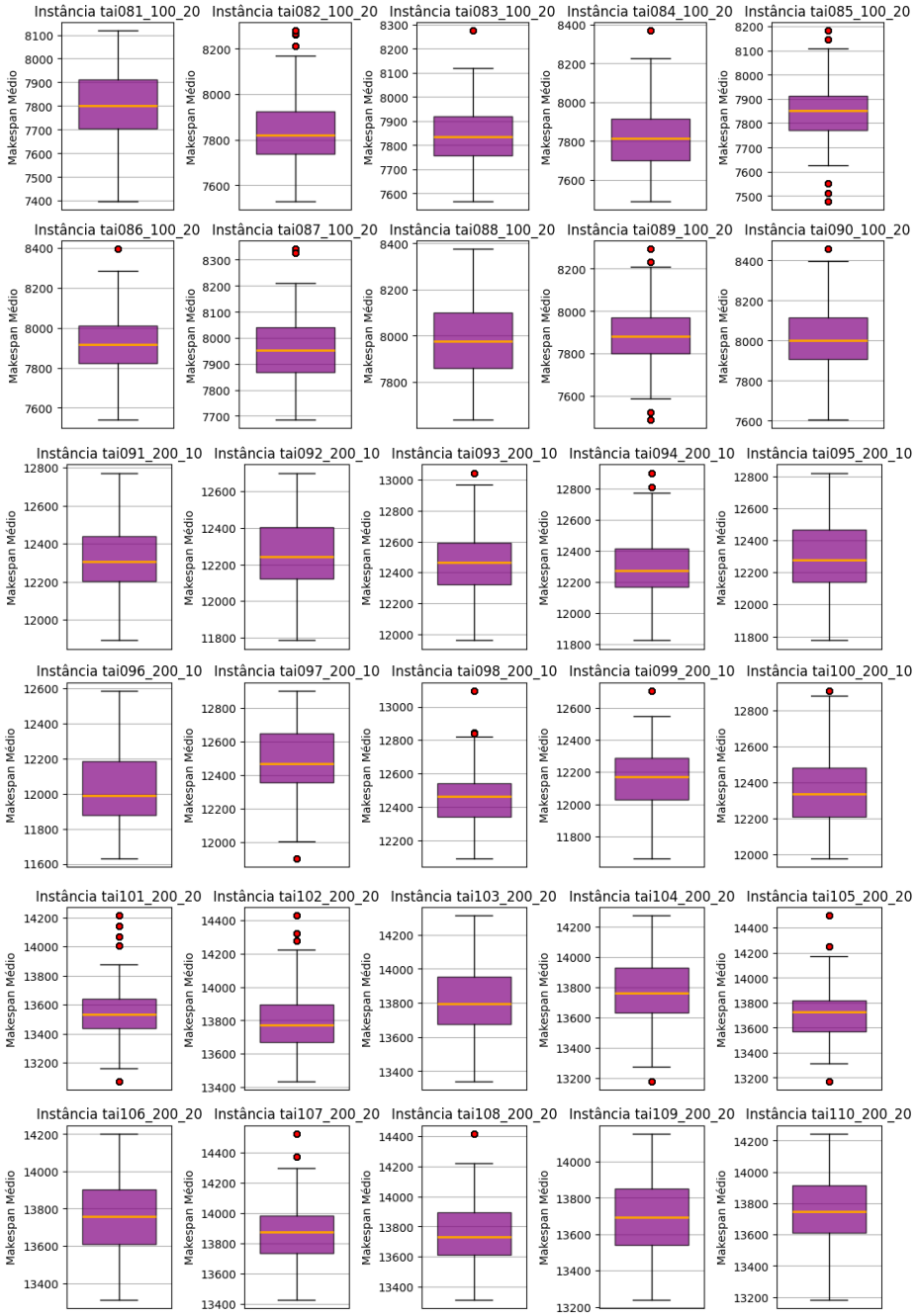
Fonte: Elaborada pelo autor (2025)

Figura 39 – Box-plot dos dados de treinamento das instâncias de tai041 a tai080



Fonte: Elaborada pelo autor (2025)

Figura 40 – Box-plot dos dados de treinamento das instâncias de tai081 a tai110



Fonte: Elaborada pelo autor (2025)

APÊNDICE B – RESULTADOS TABELADOS

Neste apêndice são apresentados os valores tabulados com todos os resultados relacionados aos métodos desenvolvidos neste trabalho.

Tabela 14 – Métricas de desempenho e Tempo de Treinamento do modelo de Regressão Linear Múltipla

Instância	R^2	$R^2(\mathbf{VC})$	**EMA**	**REMQ**	**Tempo de Treinamento (s)**
tai001_20_5	0,9372	0,9388	11,8795	14,7697	0,3008
tai002_20_5	0,9401	0,9376	11,746	14,543	0,3153
tai003_20_5	0,9587	0,9582	12,782	16,1716	0,3339
tai004_20_5	0,8805	0,8796	17,5746	21,952	0,3989
tai005_20_5	0,9585	0,9586	10,1051	12,5699	0,4356
tai006_20_5	0,9453	0,9441	12,8745	15,8956	0,3869
tai007_20_5	0,9437	0,9422	13,9672	17,3652	0,4342
tai008_20_5	0,9612	0,9618	11,2876	14,1592	0,3882
tai009_20_5	0,9624	0,9622	11,6326	14,7646	0,3913
tai010_20_5	0,9476	0,9467	14,5108	18,0688	0,3840
tai011_20_10	0,9932	0,9928	5,4054	6,7843	1,1228
tai012_20_10	0,9936	0,9937	4,8365	6,0831	1,1305
tai013_20_10	0,9951	0,995	4,574	5,7624	1,1289
tai014_20_10	0,9927	0,9927	5,2102	6,512	1,0606
tai015_20_10	0,9951	0,9948	4,6591	5,8794	1,1440
tai016_20_10	0,9945	0,9946	4,1655	5,1775	1,0893
tai017_20_10	0,9952	0,995	4,3794	5,5662	3,5622
tai018_20_10	0,9938	0,9938	4,9437	6,1962	1,5734
tai019_20_10	0,9917	0,9916	6,0019	7,5284	1,2009
tai020_20_10	0,9936	0,9935	4,6737	5,8294	1,0190
tai021_20_20	0,9977	0,9976	3,4132	4,268	3,7970
tai022_20_20	0,9973	0,9973	3,1893	3,994	2,9661
tai023_20_20	0,9974	0,9975	3,4947	4,3971	3,1833
tai024_20_20	0,9978	0,9978	3,0373	3,7764	4,6096

Continua na página seguinte

Tabela 14 – Métricas de desempenho e Tempo de Treinamento do modelo de Regressão Linear Múltipla (continuação)

Instância	R^2	R^2(**VC**)	**EMA**	**REMQ**	**Tempo de Treinamento (s)**
tai025_20_20	0,9972	0,9971	3,7676	4,7511	2,6068
tai026_20_20	0,9976	0,9976	3,2572	4,0905	2,4620
tai027_20_20	0,9977	0,9977	3,3919	4,2476	2,5399
tai028_20_20	0,9974	0,9975	3,3901	4,2796	2,7770
tai029_20_20	0,9977	0,9976	3,8485	4,8225	2,4342
tai030_20_20	0,9979	0,998	3,2129	4,0129	2,3929
tai031_50_5	0,9969	0,9969	6,0201	7,5816	1,3427
tai032_50_5	0,9964	0,9963	5,4023	6,7893	2,2492
tai033_50_5	0,9965	0,9965	5,2134	6,5895	1,3399
tai034_50_5	0,9963	0,9961	4,9025	6,1647	1,3475
tai035_50_5	0,9967	0,9968	4,5633	5,7636	1,3400
tai036_50_5	0,996	0,9961	4,7697	5,9296	1,3452
tai037_50_5	0,9958	0,9956	4,783	5,9857	1,3617
tai038_50_5	0,9967	0,9966	4,5888	5,7909	1,3607
tai039_50_5	0,9958	0,9959	5,2542	6,5268	1,3351
tai040_50_5	0,9966	0,9964	5,3818	6,6986	1,3287
tai041_50_10	0,9985	0,9985	3,772	4,7418	5,1043
tai042_50_10	0,9984	0,9984	3,4446	4,299	6,8527
tai043_50_10	0,9986	0,9987	3,343	4,2329	5,6666
tai044_50_10	0,9984	0,9983	3,8229	4,8317	4,8909
tai045_50_10	0,9986	0,9986	3,7456	4,7096	3,8979
tai046_50_10	0,9983	0,9984	3,6194	4,5377	3,9558
tai047_50_10	0,9985	0,9985	3,3155	4,1348	4,7446
tai048_50_10	0,9984	0,9985	3,5034	4,3817	4,9651
tai049_50_10	0,9985	0,9986	3,6061	4,4777	4,4363
tai050_50_10	0,9985	0,9985	3,9731	4,9574	4,6852
tai051_50_20	0,9991	0,9992	2,6433	3,315	22,8350

Continua na página seguinte

Tabela 14 – Métricas de desempenho e Tempo de Treinamento do modelo de Regressão Linear Múltipla (continuação)

Instância	R^2	R^2(**VC**)	**EMA**	**REMQ**	**Tempo de Treinamento (s)**
tai052_50_20	0,9992	0,9992	2,4968	3,1123	10,3370
tai053_50_20	0,9993	0,9993	2,1957	2,7486	10,4219
tai054_50_20	0,9992	0,9992	2,0955	2,6144	11,5706
tai055_50_20	0,9992	0,9993	2,6104	3,2709	10,1989
tai056_50_20	0,9992	0,9993	2,574	3,2003	11,4110
tai057_50_20	0,9992	0,9992	2,8129	3,5387	10,2064
tai058_50_20	0,9993	0,9993	2,5072	3,1739	10,2824
tai059_50_20	0,9993	0,9993	2,4393	3,0433	10,1749
tai060_50_20	0,9992	0,9992	2,5267	3,1728	10,3923
tai061_100_5	0,9983	0,9984	4,811	6,003	3,8838
tai062_100_5	0,9983	0,9983	5,0711	6,3239	4,3098
tai063_100_5	0,9983	0,9984	4,6558	5,8222	4,4186
tai064_100_5	0,9984	0,9985	4,7443	5,9023	4,6224
tai065_100_5	0,9983	0,9984	4,8691	6,1199	4,8994
tai066_100_5	0,9986	0,9986	5,2578	6,5291	4,9486
tai067_100_5	0,9984	0,9985	4,4221	5,5594	5,1691
tai068_100_5	0,9986	0,9986	4,5406	5,669	4,9808
tai069_100_5	0,9985	0,9985	4,5224	5,7106	4,6670
tai070_100_5	0,9987	0,9988	4,4628	5,6191	4,5604
tai071_100_10	0,9992	0,9992	3,2662	4,0936	10,6028
tai072_100_10	0,9993	0,9993	3,7221	4,6827	10,4482
tai073_100_10	0,9992	0,9992	3,3788	4,1934	10,3340
tai074_100_10	0,9991	0,9991	3,4118	4,243	10,5424
tai075_100_10	0,9992	0,9992	3,3149	4,1236	10,1934
tai076_100_10	0,9993	0,9993	2,6771	3,3702	11,1912
tai077_100_10	0,9992	0,9993	3,299	4,093	11,3656
tai078_100_10	0,9993	0,9993	3,1548	3,9471	10,7216

Continua na página seguinte

Tabela 14 – Métricas de desempenho e Tempo de Treinamento do modelo de Regressão Linear Múltipla (continuação)

Instância	R^2	R^2(**VC**)	**EMA**	**REMQ**	**Tempo de Treinamento (s)**
tai079_100_10	0,9992	0,9992	3,0011	3,7813	10,4894
tai080_100_10	0,9992	0,9992	3,2713	4,1007	10,7400
tai081_100_20	0,9995	0,9996	2,6728	3,3384	40,4193
tai082_100_20	0,9995	0,9996	2,8051	3,5184	41,3084
tai083_100_20	0,9995	0,9996	2,3164	2,9051	41,1850
tai084_100_20	0,9996	0,9996	2,5982	3,2388	41,1968
tai085_100_20	0,9995	0,9996	2,1294	2,6713	40,9258
tai086_100_20	0,9996	0,9996	2,4588	3,0857	40,2610
tai087_100_20	0,9996	0,9996	2,1557	2,6976	40,0964
tai088_100_20	0,9995	0,9996	2,7917	3,4905	40,0647
tai089_100_20	0,9995	0,9996	2,4564	3,0686	39,5044
tai090_100_20	0,9995	0,9996	2,7163	3,3911	40,1259
tai091_200_10	0,9995	0,9996	3,2821	4,1051	40,1171
tai092_200_10	0,9996	0,9996	3,3329	4,172	38,9984
tai093_200_10	0,9995	0,9996	3,995	5,0047	39,4222
tai094_200_10	0,9996	0,9996	3,34	4,1802	39,3783
tai095_200_10	0,9995	0,9996	3,8177	4,7941	39,4982
tai096_200_10	0,9996	0,9996	3,5752	4,4961	40,1433
tai097_200_10	0,9996	0,9996	3,3959	4,2611	39,0731
tai098_200_10	0,9995	0,9996	3,138	3,9207	39,4871
tai099_200_10	0,9995	0,9996	3,4225	4,3088	39,1118
tai100_200_10	0,9995	0,9996	3,3641	4,216	39,2436
tai101_200_20	0,9996	0,9997	2,8733	3,5839	231,7581
tai102_200_20	0,9996	0,9997	2,8855	3,6083	226,1561
tai103_200_20	0,9996	0,9997	3,229	4,0706	232,4809
tai104_200_20	0,9996	0,9997	3,5633	4,4555	227,9357
tai105_200_20	0,9996	0,9997	3,2769	4,0793	228,9578

Continua na página seguinte

Tabela 14 – Métricas de desempenho e Tempo de Treinamento do modelo de Regressão Linear Múltipla (continuação)

Instância	R^2	R^2(**VC**)	**EMA**	**REMQ**	**Tempo de Treinamento (s)**
tai106_200_20	0,9996	0,9997	3,3103	4,1343	226,5687
tai107_200_20	0,9996	0,9997	3,1508	3,9397	224,2672
tai108_200_20	0,9997	0,9997	3,0163	3,7744	285,9383
tai109_200_20	0,9996	0,9997	3,3273	4,1602	252,4162
tai110_200_20	0,9996	0,9997	3,5524	4,4096	240,0775

Fonte: Elaborada pelo autor (2025)

Tabela 15 – Resultados obtidos pelo método MSNEHT-RLM

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai001_20_5	1289,66	1289,33	0,07	1,12	1,20	42	10
tai002_20_5	1372,03	1372,24	0,08	1,55	1,63	53	10
tai003_20_5	1139,65	1139,40	0,08	1,63	1,71	56	10
tai004_20_5	1333,22	1333,43	0,08	1,06	1,14	36	10
tai005_20_5	1305,01	1305,72	0,09	2,01	2,09	68	10
tai006_20_5	1215,36	1215,89	0,11	2,01	2,12	52	10
tai007_20_5	1254,89	1255,30	0,08	2,10	2,19	72	10
tai008_20_5	1231,14	1230,84	0,08	1,33	1,41	46	10
tai009_20_5	1278,59	1278,38	0,08	1,13	1,21	38	10
tai010_20_5	1136,43	1136,91	0,08	2,22	2,31	75	10
tai011_20_10	1651,81	1651,19	0,16	2,19	2,35	37	10
tai012_20_10	1745,34	1745,26	0,16	2,86	3,02	48	10
tai013_20_10	1572,53	1572,23	0,16	3,27	3,44	55	10
tai014_20_10	1417,61	1417,98	0,16	3,66	3,82	61	10
tai015_20_10	1485,11	1485,58	0,16	3,17	3,33	54	10
tai016_20_10	1440,51	1440,28	0,16	4,55	4,71	77	10
tai017_20_10	1511,85	1512,67	0,16	2,69	2,85	45	10
tai018_20_10	1591,03	1591,87	0,16	3,50	3,66	58	10

Continua na página seguinte

Tabela 15 – Resultados obtidos pelo método MSNEHT-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai019_20_10	1648,67	1647,42	0,17	2,29	2,46	38	10
tai020_20_10	1640,70	1641,18	0,16	3,71	3,88	62	10
tai021_20_20	2409,86	2408,15	0,32	3,67	3,99	31	10
tai022_20_20	2167,88	2169,04	0,32	6,28	6,60	53	10
tai023_20_20	2402,08	2402,65	0,32	6,53	6,85	55	10
tai024_20_20	2271,84	2271,88	0,32	6,91	7,23	58	10
tai025_20_20	2374,11	2375,11	0,32	6,63	6,95	56	10
tai026_20_20	2298,96	2298,45	0,32	6,85	7,18	58	10
tai027_20_20	2356,25	2355,85	0,32	6,12	6,44	52	10
tai028_20_20	2277,09	2277,26	0,32	5,41	5,73	46	10
tai029_20_20	2290,04	2289,58	0,32	7,74	8,06	65	10
tai030_20_20	2258,34	2258,80	0,32	6,64	6,96	56	10
tai031_50_5	2733,22	2734,60	1,03	6,41	7,44	87	10
tai032_50_5	2852,14	2852,91	1,03	5,80	6,83	77	10
tai033_50_5	2635,18	2635,32	1,04	5,03	6,07	68	10
tai034_50_5	2791,47	2790,99	1,03	7,28	8,31	98	10
tai035_50_5	2867,04	2866,32	1,03	6,75	7,78	91	10
tai036_50_5	2843,40	2843,71	1,03	6,96	7,98	94	10
tai037_50_5	2752,83	2753,50	1,04	6,51	7,55	87	10
tai038_50_5	2699,38	2698,96	1,04	5,37	6,40	72	10
tai039_50_5	2566,61	2566,55	1,04	6,51	7,55	86	10
tai040_50_5	2788,36	2789,01	1,03	6,84	7,88	91	10
tai041_50_10	3146,99	3147,09	2,07	10,62	12,68	70	10
tai042_50_10	3017,95	3017,60	2,06	9,09	11,15	61	10
tai043_50_10	3022,76	3022,90	2,10	12,13	14,23	81	10
tai044_50_10	3163,33	3164,03	2,08	13,73	15,81	92	10
tai045_50_10	3134,32	3134,82	2,07	9,40	11,47	62	10
tai046_50_10	3151,83	3152,55	2,07	11,97	14,04	80	10

Continua na página seguinte

Tabela 15 – Resultados obtidos pelo método MSNEHT-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai047_50_10	3261,13	3260,02	2,04	8,07	10,11	54	10
tai048_50_10	3130,38	3129,86	2,08	9,94	12,02	65	10
tai049_50_10	3018,81	3019,76	2,09	13,17	15,26	86	10
tai050_50_10	3230,64	3230,24	2,17	8,29	10,46	52	10
tai051_50_20	4081,73	4082,16	4,19	23,91	28,10	79	10
tai052_50_20	3921,06	3921,66	4,17	22,85	27,01	76	10
tai053_50_20	3905,18	3905,31	4,24	28,43	32,67	94	10
tai054_50_20	3996,36	3995,12	4,18	25,16	29,35	83	10
tai055_50_20	3843,51	3844,18	4,19	24,74	28,93	82	10
tai056_50_20	3893,59	3893,28	4,15	24,89	29,05	82	10
tai057_50_20	3922,04	3922,63	4,13	27,85	31,99	91	10
tai058_50_20	3953,81	3954,24	4,15	26,34	30,49	87	10
tai059_50_20	3974,29	3973,87	4,17	26,13	30,30	86	10
tai060_50_20	4002,53	4002,47	4,18	27,05	31,23	89	10
tai061_100_5	5507,11	5507,45	7,67	13,12	20,79	88	10
tai062_100_5	5298,57	5299,09	7,64	14,43	22,07	96	10
tai063_100_5	5200,11	5199,78	7,69	17,91	25,61	118	10
tai064_100_5	5022,52	5023,85	7,76	13,88	21,64	93	10
tai065_100_5	5262,21	5264,13	7,75	16,96	24,71	113	10
tai066_100_5	5143,82	5143,71	7,67	12,81	20,48	85	10
tai067_100_5	5282,49	5282,76	7,71	16,98	24,69	114	10
tai068_100_5	5129,81	5129,54	7,93	15,92	23,86	105	10
tai069_100_5	5467,45	5468,36	7,75	16,07	23,83	106	10
tai070_100_5	5349,78	5349,99	7,67	12,31	19,99	82	10
tai071_100_10	5876,05	5876,00	15,64	22,79	38,42	74	10
tai072_100_10	5441,96	5442,65	15,70	30,38	46,08	97	10
tai073_100_10	5756,26	5756,31	15,51	35,96	51,47	117	10
tai074_100_10	5989,79	5989,40	15,63	21,33	36,96	70	10

Continua na página seguinte

Tabela 15 – Resultados obtidos pelo método MSNEHT-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai075_100_10	5636,83	5637,88	15,63	33,91	49,54	109	10
tai076_100_10	5383,97	5383,38	15,67	27,38	43,05	87	10
tai077_100_10	5721,82	5721,65	15,70	29,12	44,82	94	10
tai078_100_10	5775,02	5777,61	15,47	26,51	41,98	86	10
tai079_100_10	6022,96	6022,14	15,58	31,60	47,18	102	10
tai080_100_10	5915,89	5915,27	15,70	30,37	46,08	98	10
tai081_100_20	6556,42	6555,60	30,87	53,94	84,82	84	10
tai082_100_20	6536,40	6536,72	31,35	57,67	89,02	91	10
tai083_100_20	6617,56	6618,15	31,29	46,02	77,32	72	10
tai084_100_20	6582,16	6582,53	30,59	58,31	88,91	92	10
tai085_100_20	6661,53	6660,12	30,97	54,96	85,93	85	10
tai086_100_20	6765,28	6764,38	30,94	53,59	84,52	84	10
tai087_100_20	6621,98	6622,12	31,06	62,92	93,98	98	10
tai088_100_20	6809,54	6809,78	31,16	66,51	97,67	104	10
tai089_100_20	6660,07	6661,33	30,64	67,61	98,25	106	10
tai090_100_20	6721,97	6720,22	30,99	65,56	96,55	103	10
tai091_200_10	10968,27	10967,85	120,45	79,15	199,60	123	10
tai092_200_10	10687,85	10689,08	119,17	71,04	190,21	110	10
tai093_200_10	11076,36	11075,47	120,02	72,71	192,73	113	10
tai094_200_10	11055,84	11057,12	118,84	70,23	189,07	109	10
tai095_200_10	10677,24	10676,50	119,82	72,33	192,14	112	10
tai096_200_10	10505,56	10504,16	119,96	72,47	192,43	109	10
tai097_200_10	10998,13	10998,77	119,84	73,52	193,36	114	10
tai098_200_10	10855,14	10854,92	119,00	71,35	190,35	110	10
tai099_200_10	10582,72	10583,70	118,21	60,76	178,97	94	10
tai100_200_10	10840,11	10841,56	118,95	73,20	192,15	112	10
tai101_200_20	11689,07	11689,99	245,54	132,39	377,93	101	10
tai102_200_20	11743,77	11744,26	240,12	172,86	412,99	132	10

Continua na página seguinte

Tabela 15 – Resultados obtidos pelo método MSNEHT-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai103_200_20	11873,85	11872,61	240,52	166,74	407,25	126	10
tai104_200_20	11768,95	11770,14	243,06	148,19	391,25	113	10
tai105_200_20	11681,47	11682,56	241,40	153,23	394,64	116	10
tai106_200_20	11680,27	11681,41	242,27	148,88	391,15	113	10
tai107_200_20	11863,71	11863,76	240,91	161,43	402,34	122	10
tai108_200_20	11887,75	11885,60	240,28	138,11	378,39	105	10
tai109_200_20	11718,45	11719,81	241,92	173,36	415,27	132	10
tai110_200_20	11784,92	11783,30	244,51	164,37	408,87	123	10

Fonte: Elaborada pelo autor (2025)

Tabela 16 – Resultados obtidos pelo método GRASP-RLM

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai001_20_5	1365,60	1365,80	3,12	9,82	12,94	56	10
tai002_20_5	1426,18	1426,36	3,26	12,84	16,11	73	10
tai003_20_5	1430,73	1431,09	3,21	10,13	13,34	53	10
tai004_20_5	1544,06	1543,77	3,55	11,02	14,57	58	10
tai005_20_5	1379,15	1379,49	3,57	14,49	18,06	76	10
tai006_20_5	1389,45	1388,97	3,59	11,26	14,85	59	10
tai007_20_5	1371,49	1371,72	3,51	10,09	13,60	53	10
tai008_20_5	1458,56	1459,13	3,58	11,84	15,42	62	10
tai009_20_5	1518,28	1519,00	3,60	10,69	14,29	56	10
tai010_20_5	1262,04	1261,98	3,55	14,35	17,90	75	10
tai011_20_10	1931,85	1931,38	6,94	22,51	29,45	59	10
tai012_20_10	1970,00	1971,55	6,94	22,14	29,09	58	10
tai013_20_10	1804,05	1804,15	6,90	31,92	38,82	84	10
tai014_20_10	1703,92	1704,33	7,00	20,57	27,57	54	10
tai015_20_10	1762,09	1761,93	6,94	20,25	27,19	53	10
tai016_20_10	1721,70	1721,79	7,05	31,24	38,29	77	10

Continua na página seguinte

Tabela 16 – Resultados obtidos pelo método GRASP-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai017_20_10	1752,68	1752,56	7,49	23,02	30,50	53	10
tai018_20_10	1825,38	1825,54	6,96	23,54	30,49	62	10
tai019_20_10	1852,44	1852,35	6,93	23,10	30,03	61	10
tai020_20_10	1848,98	1848,52	6,90	26,92	33,83	70	10
tai021_20_20	2686,08	2685,14	13,51	43,15	56,66	57	10
tai022_20_20	2498,79	2498,22	13,64	43,52	57,16	57	10
tai023_20_20	2610,91	2610,24	13,72	41,96	55,68	55	10
tai024_20_20	2647,88	2647,74	13,64	38,21	51,85	50	10
tai025_20_20	2675,52	2675,17	13,51	39,20	52,71	51	10
tai026_20_20	2549,17	2548,77	13,61	50,67	64,27	67	10
tai027_20_20	2529,71	2530,88	13,85	51,84	65,69	68	10
tai028_20_20	2505,24	2505,96	13,65	38,69	52,33	51	10
tai029_20_20	2514,95	2515,02	13,67	47,90	61,58	63	10
tai030_20_20	2587,89	2587,93	13,55	43,14	56,69	57	10
tai031_50_5	3048,34	3047,65	9,14	51,18	60,31	109	10
tai032_50_5	3384,38	3386,17	9,47	46,66	56,14	98	10
tai033_50_5	3109,71	3110,04	9,42	40,55	49,97	85	10
tai034_50_5	3243,79	3244,08	9,31	42,98	52,30	91	10
tai035_50_5	3334,70	3334,16	9,24	47,36	56,61	101	10
tai036_50_5	3230,03	3229,78	9,24	44,07	53,31	94	10
tai037_50_5	3171,73	3172,34	9,23	37,07	46,30	79	10
tai038_50_5	3085,66	3086,80	9,32	43,59	52,91	92	10
tai039_50_5	2842,14	2842,66	9,31	35,54	44,85	75	10
tai040_50_5	3271,23	3270,95	9,17	39,70	48,88	85	10
tai041_50_10	3728,48	3728,66	18,43	75,31	93,74	80	10
tai042_50_10	3632,33	3631,69	18,05	62,61	80,66	66	10
tai043_50_10	3688,22	3688,90	18,37	79,09	97,46	84	10
tai044_50_10	3782,24	3781,04	18,31	70,76	89,07	75	10

Continua na página seguinte

Tabela 16 – Resultados obtidos pelo método GRASP-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai045_50_10	3819,07	3819,27	18,33	96,41	114,74	102	10
tai046_50_10	3681,17	3682,41	18,40	58,82	77,22	62	10
tai047_50_10	3753,12	3754,25	18,15	86,94	105,09	93	10
tai048_50_10	3734,52	3735,98	18,20	70,23	88,44	75	10
tai049_50_10	3677,10	3677,39	18,32	87,99	106,31	93	10
tai050_50_10	3802,50	3802,72	18,54	84,16	102,70	89	10
tai051_50_20	4781,14	4781,98	36,50	138,46	174,96	73	10
tai052_50_20	4601,65	4602,21	36,55	147,93	184,48	78	10
tai053_50_20	4553,57	4553,57	36,21	182,88	219,10	97	10
tai054_50_20	4654,45	4652,62	36,52	177,13	213,66	93	10
tai055_50_20	4714,98	4715,06	36,64	163,22	199,86	86	10
tai056_50_20	4712,18	4714,28	37,50	196,87	234,37	103	10
tai057_50_20	4778,30	4779,02	36,53	181,20	217,74	96	10
tai058_50_20	4769,72	4769,10	36,61	162,98	199,59	86	10
tai059_50_20	4731,67	4732,71	36,55	142,28	178,82	75	10
tai060_50_20	4834,09	4833,56	37,31	134,79	172,10	71	10
tai061_100_5	6562,30	6563,26	23,78	88,85	112,63	94	10
tai062_100_5	6319,19	6319,65	23,83	95,90	119,73	102	10
tai063_100_5	6075,12	6076,47	24,11	112,77	136,89	120	10
tai064_100_5	5810,82	5809,85	23,89	112,79	136,68	120	10
tai065_100_5	6144,15	6144,71	23,76	98,53	122,29	105	10
tai066_100_5	6157,71	6160,42	23,77	96,79	120,55	103	10
tai067_100_5	6121,89	6123,96	23,88	96,62	120,50	103	10
tai068_100_5	6226,56	6227,44	23,93	109,56	133,49	116	10
tai069_100_5	6408,26	6407,56	23,73	90,28	114,00	96	10
tai070_100_5	6321,32	6319,72	23,85	118,69	142,54	126	10
tai071_100_10	7058,10	7057,26	47,22	191,12	238,34	101	10
tai072_100_10	6765,94	6765,70	47,43	211,94	259,37	112	10

Continua na página seguinte

Tabela 16 – Resultados obtidos pelo método GRASP-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai073_100_10	6770,09	6768,18	48,07	184,92	232,99	97	10
tai074_100_10	7191,89	7190,70	47,60	153,11	200,70	81	10
tai075_100_10	6781,42	6782,05	47,03	190,25	237,28	102	10
tai076_100_10	6709,60	6708,35	47,47	170,82	218,30	91	10
tai077_100_10	6673,31	6673,10	47,54	160,45	207,98	85	10
tai078_100_10	6939,86	6941,02	47,45	200,30	247,75	106	10
tai079_100_10	6948,49	6948,79	47,39	193,82	241,22	100	10
tai080_100_10	7012,92	7011,21	47,12	164,20	211,33	88	10
tai081_100_20	7826,05	7825,04	94,76	335,68	430,44	89	10
tai082_100_20	7921,04	7920,53	94,57	445,78	540,36	118	10
tai083_100_20	7952,01	7951,46	94,73	386,11	480,84	102	10
tai084_100_20	7932,54	7930,13	94,52	324,64	419,16	86	10
tai085_100_20	8131,19	8131,23	94,65	445,45	540,10	118	10
tai086_100_20	7977,18	7976,73	93,95	361,45	455,40	96	10
tai087_100_20	8045,71	8046,00	95,22	377,49	472,71	99	10
tai088_100_20	8134,12	8133,17	95,05	381,97	477,02	100	10
tai089_100_20	7802,46	7802,03	94,64	374,72	469,36	99	10
tai090_100_20	8095,28	8094,87	94,39	352,59	446,99	93	10
tai091_200_10	12759,40	12760,15	183,16	478,03	661,18	126	10
tai092_200_10	12955,89	12959,65	185,12	494,87	679,98	130	10
tai093_200_10	13190,63	13190,09	182,82	492,85	675,67	130	10
tai094_200_10	12884,73	12886,44	182,45	432,27	614,72	115	10
tai095_200_10	13014,80	13015,59	182,62	403,17	585,79	107	10
tai096_200_10	12669,54	12668,62	182,46	507,34	689,80	134	10
tai097_200_10	13324,30	13325,50	182,84	405,59	588,42	108	10
tai098_200_10	13315,83	13316,57	181,63	401,26	582,89	106	10
tai099_200_10	12818,63	12817,24	183,20	391,48	574,67	103	10
tai100_200_10	13159,03	13158,48	182,72	350,65	533,37	93	10

Continua na página seguinte

Tabela 16 – Resultados obtidos pelo método GRASP-RLM (Continuação)

Instância	**MM (*pool*)**	**MMC (*pool*)**	**T(s)**	**T *pool* (s)**	**TT (s)**	**QS**	**Nº It.**
tai101_200_20	14151,44	14150,95	368,94	887,71	1256,64	117	10
tai102_200_20	14239,89	14239,60	366,97	932,83	1299,81	124	10
tai103_200_20	14471,90	14470,77	368,10	716,97	1085,07	95	10
tai104_200_20	13755,67	13757,92	369,24	919,22	1288,46	121	10
tai105_200_20	14186,34	14183,74	371,86	961,90	1333,76	126	10
tai106_200_20	14260,49	14260,90	367,69	864,24	1231,93	115	10
tai107_200_20	14381,56	14382,92	371,20	901,56	1272,76	116	10
tai108_200_20	14271,98	14270,94	369,71	902,94	1272,65	119	10
tai109_200_20	14179,21	14178,50	367,46	886,73	1254,20	117	10
tai110_200_20	14247,80	14248,16	367,84	875,71	1243,55	116	10

Fonte: Elaborada pelo autor (2025)

Tabela 17 – Resultados obtidos pelos métodos simheurísticos MSNEHT-SIMH e GRASP-SIMH

Instância	**MSNEHT-SIMH**				**GRASP-SIMH**			
	MM	**MMC**	**T(s)**	**Nº It.**	**MM**	**MMC**	**T(s)**	**Nº It.**
tai001_20_5	1284,58	1288,17	18,67	10	1358,61	1466,96	18,58	10
tai002_20_5	1366,26	1370,20	18,62	10	1421,60	1526,84	18,48	10
tai003_20_5	1133,26	1153,51	20,17	10	1313,41	1414,97	20,17	10
tai004_20_5	1309,57	1309,44	20,43	10	1518,84	1666,42	20,08	10
tai005_20_5	1276,03	1286,97	20,10	10	1381,00	1501,14	20,02	10
tai006_20_5	1216,63	1217,49	20,35	10	1366,98	1493,75	19,98	10
tai007_20_5	1250,71	1259,85	20,08	10	1377,14	1458,26	20,04	10
tai008_20_5	1217,91	1226,86	20,34	10	1329,30	1467,19	20,20	10
tai009_20_5	1261,69	1264,23	20,37	10	1471,96	1579,21	20,03	10
tai010_20_5	1127,33	1128,34	20,19	10	1274,09	1394,83	20,29	10
tai011_20_10	1616,87	1636,72	41,45	10	1819,33	1982,53	40,81	10
tai012_20_10	1730,02	1739,22	41,43	10	1919,68	2076,16	40,83	10
tai013_20_10	1554,19	1557,71	41,41	10	1729,94	1862,32	40,80	10

Continua na página seguinte

Tabela 17 – Resultados obtidos pelos métodos simheurísticos MSNEHT-SIMH e GRASP-SIMH (Continuação)

Instância	MSNEHT-SIMH				GRASP-SIMH			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai014_20_10	1411,27	1440,20	41,25	10	1610,61	1706,16	40,67	10
tai015_20_10	1467,14	1467,53	40,63	10	1746,06	1905,02	40,41	10
tai016_20_10	1439,74	1449,04	41,54	10	1758,47	1901,14	40,19	10
tai017_20_10	1510,20	1513,74	41,31	10	1655,68	1774,85	40,08	10
tai018_20_10	1599,49	1610,94	41,62	10	1826,48	1914,20	40,51	10
tai019_20_10	1635,70	1637,27	41,20	10	1841,93	2013,54	40,24	10
tai020_20_10	1631,99	1648,72	41,14	10	1890,99	2000,86	40,29	10
tai021_20_20	2335,03	2339,90	86,25	10	2649,92	2882,27	84,45	10
tai022_20_20	2152,56	2168,62	86,32	10	2548,61	2736,79	83,70	10
tai023_20_20	2387,39	2388,16	86,43	10	2605,42	2718,83	84,31	10
tai024_20_20	2278,82	2279,26	85,41	10	2531,57	2633,77	84,08	10
tai025_20_20	2354,52	2361,65	85,48	10	2640,15	2778,55	84,52	10
tai026_20_20	2252,95	2267,88	87,82	10	2497,53	2587,36	84,12	10
tai027_20_20	2333,66	2336,72	86,65	10	2568,36	2704,53	84,54	10
tai028_20_20	2236,10	2262,40	86,91	10	2535,71	2733,34	84,86	10
tai029_20_20	2291,71	2318,99	86,41	10	2564,49	2730,51	84,31	10
tai030_20_20	2233,96	2243,38	85,18	10	2564,16	2740,92	84,57	10
tai031_50_5	2730,24	2733,51	325,34	10	3033,39	3125,59	323,27	10
tai032_50_5	2843,17	2848,00	319,01	10	3297,63	3455,35	319,82	10
tai033_50_5	2630,08	2632,86	330,90	10	3115,13	3278,38	322,72	10
tai034_50_5	2773,78	2780,94	328,55	10	3296,93	3464,71	324,02	10
tai035_50_5	2865,93	2871,24	321,81	10	3330,04	3470,56	324,87	10
tai036_50_5	2837,35	2845,03	328,54	10	3138,65	3278,14	328,55	10
tai037_50_5	2742,53	2744,56	331,85	10	3117,08	3243,89	325,18	10
tai038_50_5	2696,18	2700,99	323,30	10	3076,70	3236,99	323,67	10
tai039_50_5	2575,75	2600,79	325,42	10	2875,85	2959,24	322,01	10
tai040_50_5	2786,44	2793,67	323,50	10	3237,31	3373,28	323,71	10

Continua na página seguinte

Tabela 17 – Resultados obtidos pelos métodos simheurísticos MSNEHT-SIMH e GRASP-SIMH (Continuação)

Instância	MSNEHT-SIMH				GRASP-SIMH			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai041_50_10	3135,10	3145,14	717,44	10	3734,82	3871,77	701,74	10
tai042_50_10	2992,53	3002,78	726,73	10	3645,68	3798,82	704,47	10
tai043_50_10	2985,62	3003,88	717,02	10	3591,28	3725,35	712,19	10
tai044_50_10	3138,78	3157,19	713,70	10	3694,95	3842,57	706,45	10
tai045_50_10	3130,59	3166,44	711,78	10	3776,98	3915,89	704,34	10
tai046_50_10	3117,39	3142,12	711,36	10	3697,40	3814,48	704,21	10
tai047_50_10	3241,93	3278,43	719,50	10	3722,24	3886,46	702,81	10
tai048_50_10	3143,71	3163,96	725,98	10	3617,77	3766,27	710,00	10
tai049_50_10	3001,87	3009,33	715,62	10	3654,16	3829,44	708,59	10
tai050_50_10	3188,66	3208,19	720,27	10	3696,58	3879,85	709,40	10
tai051_50_20	4038,63	4048,79	1530,00	10	4712,71	4863,89	1507,85	10
tai052_50_20	3896,43	3913,22	1529,60	10	4533,09	4680,94	1507,41	10
tai053_50_20	3872,99	3882,05	1531,31	10	4460,02	4605,90	1515,09	10
tai054_50_20	3932,65	3942,48	1515,71	10	4560,82	4695,70	1511,59	10
tai055_50_20	3813,47	3823,08	1518,49	10	4631,38	4766,49	1514,36	10
tai056_50_20	3852,57	3859,21	1520,82	10	4627,61	4755,25	1496,10	10
tai057_50_20	3900,29	3918,20	1531,14	10	4658,58	4788,56	1507,66	10
tai058_50_20	3879,97	3894,67	1511,07	10	4699,85	4893,98	1517,19	10
tai059_50_20	3942,34	3960,94	1516,41	10	4594,59	4704,19	1504,07	10
tai060_50_20	3968,14	3975,98	1530,30	10	4635,68	4771,26	1513,54	10
tai061_100_5	5496,73	5503,50	2898,35	10	6423,57	6595,62	2811,47	10
tai062_100_5	5283,38	5285,11	2848,35	10	6190,91	6362,63	2782,67	10
tai063_100_5	5207,02	5212,57	2828,33	10	6065,93	6202,52	2807,63	10
tai064_100_5	5021,44	5025,16	2872,81	10	5871,42	6040,26	2823,10	10
tai065_100_5	5266,68	5269,98	2806,69	10	6045,78	6195,51	2805,63	10
tai066_100_5	5141,68	5140,86	2842,75	10	5958,80	6136,47	2819,31	10
tai067_100_5	5269,46	5286,70	2858,95	10	6132,68	6325,06	2806,85	10

Continua na página seguinte

Tabela 17 – Resultados obtidos pelos métodos simheurísticos MSNEHT-SIMH e GRASP-SIMH (Continuação)

Instância	MSNEHT-SIMH				GRASP-SIMH			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai068_100_5	5118,80	5144,28	2814,24	10	6106,63	6269,70	2803,83	10
tai069_100_5	5471,65	5510,31	2868,45	10	6390,58	6549,63	2807,38	10
tai070_100_5	5335,84	5363,55	2873,17	10	6326,39	6508,24	2824,86	10
tai071_100_10	5861,14	5871,65	3600,19	6	7105,48	7270,93	3600,22	6
tai072_100_10	5428,23	5448,06	3600,14	6	6747,24	6917,95	3600,20	6
tai073_100_10	5741,36	5755,12	3600,24	6	6741,85	6922,75	3600,22	6
tai074_100_10	5945,58	5967,05	3600,24	6	7157,64	7318,31	3600,23	6
tai075_100_10	5602,70	5612,25	3600,15	6	6800,79	6964,49	3600,22	6
tai076_100_10	5369,44	5385,45	3600,14	6	6705,93	6869,11	3600,14	6
tai077_100_10	5697,05	5710,55	3600,16	6	6617,31	6817,85	3600,22	6
tai078_100_10	5751,83	5764,19	3600,19	6	6950,81	7099,53	3600,19	6
tai079_100_10	6002,07	6042,10	3600,24	6	7120,95	7276,03	3600,23	6
tai080_100_10	5908,75	5930,11	3600,17	6	6970,20	7109,20	3600,20	6
tai081_100_20	6583,35	6607,31	3600,43	3	7908,82	8045,68	3600,52	3
tai082_100_20	6497,34	6518,20	3600,42	3	7920,11	8082,04	3600,42	3
tai083_100_20	6613,19	6620,11	3600,33	3	7748,58	7903,72	3600,31	3
tai084_100_20	6609,24	6620,02	3600,30	3	8096,27	8266,20	3600,45	3
tai085_100_20	6649,10	6669,49	3600,39	3	8076,48	8231,68	3600,37	3
tai086_100_20	6720,04	6739,40	3600,38	3	8025,35	8199,92	3600,27	3
tai087_100_20	6644,48	6682,35	3600,29	3	7947,12	8122,59	3600,45	3
tai088_100_20	6789,55	6803,64	3600,35	3	8154,81	8333,52	3600,40	3
tai089_100_20	6684,92	6712,49	3600,42	3	7982,68	8148,76	3600,51	3
tai090_100_20	6713,49	6729,16	3600,34	3	7955,87	8117,29	3600,49	3
tai091_100_20	6644,92	6678,42	3600,34	3	7951,73	8116,29	3600,38	3
tai092_100_20	6772,34	6798,18	3600,28	3	8031,11	8198,20	3600,27	3
tai093_100_20	6798,84	6817,71	3600,42	3	8047,69	8213,73	3600,33	3
tai094_100_20	6862,73	6884,13	3600,45	3	8189,85	8363,56	3600,37	3

Continua na página seguinte

Tabela 17 – Resultados obtidos pelos métodos simheurísticos MSNEHT-SIMH e GRASP-SIMH (Continuação)

Instância	MSNEHT-SIMH				GRASP-SIMH			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai095_100_20	6881,32	6902,52	3600,38	3	8186,27	8352,83	3600,32	3
tai096_100_20	6734,13	6758,88	3600,37	3	7956,77	8125,28	3600,32	3
tai097_100_20	6756,49	6777,51	3600,34	3	7983,72	8147,23	3600,36	3
tai098_100_20	6811,16	6832,79	3600,40	3	8065,55	8233,91	3600,35	3
tai099_100_20	6824,13	6843,24	3600,33	3	8110,43	8284,65	3600,41	3
tai100_100_20	6869,11	6890,92	3600,28	3	8181,72	8349,18	3600,32	3
tai101_200_10	11719,75	11768,48	3600,79	1	14149,64	14289,43	3600,57	1
tai102_200_10	11846,91	11900,73	3600,71	1	14844,91	15015,14	3600,83	1
tai103_200_10	11954,99	11989,28	3601,03	1	14430,70	14610,52	3600,57	1
tai104_200_10	11813,65	11845,74	3600,63	1	14687,03	14867,52	3600,88	1
tai105_200_10	11775,71	11787,96	3600,76	1	14544,28	14710,34	3600,79	1
tai106_200_10	11778,70	11814,26	3600,87	1	15091,28	15287,29	3600,68	1
tai107_200_10	11882,99	11912,77	3600,86	1	14456,27	14639,49	3601,02	1
tai108_200_10	11875,06	11900,45	3600,66	1	14741,48	14926,70	3600,79	1
tai109_200_10	11700,20	11707,47	3600,97	1	14555,41	14722,45	3600,90	1
tai110_200_10	11864,38	11889,12	3600,77	1	14313,03	14486,75	3600,73	1

Fonte: Elaborada pelo autor (2025)

Tabela 18 – Resultados obtidos pelos métodos simheurísticos MSIQL-S e MSQLNEH-S

Instância	MSIQL-S				MSQLNEH-S			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai001_20_5	1289,07	1296,97	21,57	10	1293,12	1297,83	21,76	10
tai002_20_5	1362,94	1368,83	22,86	10	1363,39	1368,00	23,47	10
tai003_20_5	1095,92	1099,63	23,54	10	1096,38	1104,91	23,25	10
tai004_20_5	1310,00	1336,49	23,26	10	1316,50	1332,06	23,22	10
tai005_20_5	1250,73	1252,25	23,51	10	1245,49	1250,48	23,57	10
tai006_20_5	1210,84	1212,82	23,45	10	1214,08	1216,70	23,55	10

Continua na página seguinte

Tabela 18 – Resultados obtidos pelos métodos simheurísticos MSIQL-S e MSQLNEH-S (Continuação)

Instância	MSIQL-S				MSQLNEH-S			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai007_20_5	1248,20	1252,60	23,48	10	1250,88	1254,59	23,61	10
tai008_20_5	1221,83	1225,02	23,23	10	1227,87	1236,60	23,80	10
tai009_20_5	1251,82	1260,35	23,30	10	1256,12	1257,74	23,44	10
tai010_20_5	1121,57	1127,61	23,58	10	1123,92	1123,63	23,50	10
tai011_20_10	1615,33	1626,06	45,11	10	1617,77	1631,08	45,10	10
tai012_20_10	1708,02	1722,36	45,15	10	1718,13	1729,01	45,27	10
tai013_20_10	1544,33	1548,25	45,49	10	1545,57	1550,76	45,45	10
tai014_20_10	1408,53	1411,02	45,35	10	1412,03	1437,44	45,14	10
tai015_20_10	1462,74	1487,59	45,03	10	1459,77	1502,44	45,01	10
tai016_20_10	1437,19	1440,60	45,43	10	1438,81	1451,25	45,63	10
tai017_20_10	1508,49	1522,42	45,25	10	1516,54	1521,28	45,23	10
tai018_20_10	1578,92	1583,69	45,19	10	1579,08	1590,92	44,93	10
tai019_20_10	1637,38	1641,54	45,90	10	1625,60	1640,23	45,39	10
tai020_20_10	1640,58	1647,55	45,48	10	1636,79	1644,84	44,92	10
tai021_20_20	2350,41	2370,56	89,43	10	2349,50	2373,85	88,01	10
tai022_20_20	2156,49	2158,00	88,52	10	2144,79	2160,62	88,89	10
tai023_20_20	2388,54	2417,05	89,39	10	2382,22	2406,72	89,44	10
tai024_20_20	2255,87	2277,97	88,53	10	2269,90	2296,28	88,07	10
tai025_20_20	2338,39	2342,90	89,00	10	2368,25	2385,43	88,05	10
tai026_20_20	2294,43	2305,84	89,41	10	2279,96	2292,82	88,65	10
tai027_20_20	2330,24	2336,04	90,45	10	2330,51	2349,52	89,15	10
tai028_20_20	2246,14	2257,60	88,36	10	2256,63	2268,81	88,08	10
tai029_20_20	2280,18	2283,85	89,47	10	2284,10	2295,13	88,26	10
tai030_20_20	2229,96	2249,85	89,64	10	2238,81	2242,71	89,58	10
tai031_50_5	2729,30	2751,74	350,99	10	2727,37	2731,54	359,27	10
tai032_50_5	2851,86	2862,23	354,72	10	2851,50	2856,81	359,38	10
tai033_50_5	2646,91	2662,21	352,26	10	2655,49	2667,94	352,10	10
tai034_50_5	2783,59	2787,52	349,61	10	2777,57	2784,56	356,33	10

Continua na página seguinte

Tabela 18 – Resultados obtidos pelos métodos simheurísticos MSIQL-S e MSQLNEH-S (Continuação)

Instância	MSIQL-S				MSQLNEH-S			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai035_50_5	2862,07	2865,08	359,07	10	2861,38	2869,70	358,66	10
tai036_50_5	2845,46	2862,56	354,05	10	2839,94	2861,55	360,22	10
tai037_50_5	2760,98	2766,77	351,26	10	2758,42	2773,18	349,96	10
tai038_50_5	2717,85	2737,56	351,93	10	2711,69	2718,85	359,12	10
tai039_50_5	2581,24	2585,80	350,52	10	2580,67	2588,83	353,59	10
tai040_50_5	2782,93	2788,07	355,21	10	2777,36	2786,51	357,75	10
tai041_50_10	3140,24	3176,65	714,52	10	3148,85	3171,17	703,35	10
tai042_50_10	3034,99	3057,24	718,97	10	3040,60	3045,89	714,35	10
tai043_50_10	3010,64	3026,22	718,05	10	2999,85	3008,41	712,30	10
tai044_50_10	3182,29	3209,38	715,21	10	3166,49	3188,06	704,76	10
tai045_50_10	3123,10	3156,09	713,98	10	3117,87	3140,14	723,92	10
tai046_50_10	3142,30	3151,22	711,87	10	3141,84	3158,23	703,08	10
tai047_50_10	3229,36	3234,06	712,19	10	3223,55	3250,79	711,10	10
tai048_50_10	3149,64	3168,52	712,29	10	3155,81	3175,14	710,59	10
tai049_50_10	3028,34	3045,89	713,19	10	3039,06	3057,03	713,82	10
tai050_50_10	3218,78	3228,02	716,14	10	3218,43	3238,15	712,56	10
tai051_50_20	4088,54	4102,49	1458,33	10	4073,00	4095,98	1469,20	10
tai052_50_20	3938,98	3986,77	1463,65	10	3925,21	3946,89	1437,22	10
tai053_50_20	3880,85	3883,39	1463,08	10	3865,64	3902,08	1464,58	10
tai054_50_20	3942,09	3998,05	1450,65	10	3939,29	3963,40	1446,89	10
tai055_50_20	3841,05	3853,74	1464,26	10	3867,24	3879,23	1475,52	10
tai056_50_20	3918,07	3948,23	1447,29	10	3888,87	3906,77	1456,15	10
tai057_50_20	3948,33	3970,93	1459,25	10	3943,13	3984,64	1460,06	10
tai058_50_20	3906,70	3915,49	1466,78	10	3926,29	3959,02	1443,69	10
tai059_50_20	3986,33	3992,68	1463,18	10	3993,90	4018,95	1466,04	10
tai060_50_20	3952,16	3964,71	1477,54	10	3985,15	4012,43	1450,09	10
tai061_100_5	5505,68	5517,02	2880,42	10	5500,79	5509,70	2853,32	10
tai062_100_5	5286,30	5307,24	2895,35	10	5287,02	5295,97	2866,91	10

Continua na página seguinte

Tabela 18 – Resultados obtidos pelos métodos simheurísticos MSIQL-S e MSQLNEH-S (Continuação)

Instância	MSIQL-S				MSQLNEH-S			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai063_100_5	5214,22	5223,34	2843,35	10	5222,64	5236,77	2894,14	10
tai064_100_5	5038,11	5048,58	2830,96	10	5030,47	5046,94	2848,39	10
tai065_100_5	5267,10	5280,85	2857,30	10	5271,09	5280,84	2874,94	10
tai066_100_5	5152,01	5163,88	2834,74	10	5147,45	5152,65	2852,84	10
tai067_100_5	5274,08	5289,41	2833,62	10	5273,81	5294,59	2862,40	10
tai068_100_5	5131,26	5143,13	2865,71	10	5131,01	5138,75	2856,66	10
tai069_100_5	5484,42	5493,26	2895,08	10	5477,10	5487,53	2841,72	10
tai070_100_5	5345,22	5351,95	2828,68	10	5345,40	5369,34	2888,35	10
tai071_100_10	5901,37	5957,24	3600,08	6	5928,56	5962,10	3600,10	6
tai072_100_10	5503,14	5553,21	3600,12	6	5485,44	5510,42	3600,00	6
tai073_100_10	5778,23	5791,46	3600,10	6	5802,86	5828,45	3600,04	6
tai074_100_10	6017,25	6031,85	3600,11	6	6019,91	6055,40	3600,05	6
tai075_100_10	5646,11	5685,70	3600,11	6	5671,89	5725,34	3600,01	6
tai076_100_10	5425,88	5460,65	3600,11	6	5413,51	5426,09	3600,04	6
tai077_100_10	5727,71	5741,91	3600,08	6	5738,75	5756,90	3600,03	6
tai078_100_10	5771,87	5806,31	3600,09	6	5793,36	5823,06	3600,10	6
tai079_100_10	6004,77	6013,62	3600,09	6	6010,18	6019,65	3600,04	6
tai080_100_10	5937,13	5945,09	3600,01	6	5932,29	5972,02	3600,05	6
tai081_100_20	6648,27	6673,69	3600,20	3	6657,47	6718,39	3600,03	3
tai082_100_20	6608,82	6625,98	3600,11	3	6651,19	6689,92	3600,01	3
tai083_100_20	6664,51	6701,49	3600,17	3	6700,17	6730,75	3600,21	3
tai084_100_20	6636,99	6658,88	3600,09	3	6626,59	6633,44	3600,18	3
tai085_100_20	6713,20	6727,77	3600,12	3	6770,54	6788,97	3600,10	3
tai086_100_20	6824,66	6872,18	3600,00	3	6740,69	6761,21	3600,24	3
tai087_100_20	6715,80	6750,62	3600,18	3	6665,91	6690,05	3600,10	3
tai088_100_20	6890,41	6908,24	3600,11	3	6853,89	6869,44	3600,16	3
tai089_100_20	6697,64	6705,08	3600,00	3	6708,55	6740,88	3600,00	3
tai090_100_20	6812,88	6844,56	3600,05	3	6835,86	6858,01	3600,01	3

Continua na página seguinte

Tabela 18 – Resultados obtidos pelos métodos simheurísticos MSIQL-S e MSQLNEH-S (Continuação)

Instância	MSIQL-S				MSQLNEH-S			
	MM	MMC	T(s)	Nº It.	MM	MMC	T(s)	Nº It.
tai091_200_10	11015,10	11016,79	3600,07	1	11029,36	11044,10	3600,06	1
tai092_200_10	10744,33	10790,39	3600,09	1	10859,82	10883,07	3600,10	1
tai093_200_10	11143,37	11157,75	3600,05	1	11193,39	11213,93	3600,01	1
tai094_200_10	11016,53	11034,95	3600,10	1	11015,56	11056,65	3600,02	1
tai095_200_10	10744,98	10764,14	3600,09	1	10762,88	10809,62	3600,05	1
tai096_200_10	10579,39	10605,89	3600,06	1	10626,42	10644,87	3600,22	1
tai097_200_10	11076,39	11097,20	3600,04	1	11120,86	11139,45	3600,04	1
tai098_200_10	10947,81	10977,23	3600,04	1	10985,29	11020,91	3600,01	1
tai099_200_10	10662,50	10719,54	3600,06	1	10634,11	10671,50	3600,14	1
tai100_200_10	10930,76	10977,70	3600,02	1	10863,19	10906,47	3600,03	1
tai101_200_20	11875,00	11896,08	3600,35	1	11860,92	11904,24	3600,02	1
tai102_200_20	11994,82	12036,14	3600,03	1	11974,48	11998,45	3600,39	1
tai103_200_20	12029,18	12068,51	3600,50	1	12016,62	12049,43	3600,35	1
tai104_200_20	11975,83	12011,74	3600,39	1	11977,65	12019,27	3600,58	1
tai105_200_20	11894,03	11918,48	3600,03	1	11882,47	11940,39	3600,46	1
tai106_200_20	11896,97	11919,33	3600,29	1	11826,83	11889,42	3600,30	1
tai107_200_20	12021,04	12028,60	3600,41	1	12088,44	12114,07	3600,21	1
tai108_200_20	12080,81	12101,17	3600,09	1	12038,80	12057,69	3600,40	1
tai109_200_20	11863,93	11904,70	3600,34	1	11871,69	11915,08	3600,39	1
tai110_200_20	12006,19	12038,17	3600,26	1	12041,29	12067,18	3600,01	1

Fonte: Elaborada pelo autor (2025)

Tabela 19 – Resultados obtidos pelos métodos QLNEH-S e IQL-S

Instância	QLNEH-S			IQL-S		
	MM	MMC	T(s)	MM	MMC	T(s)
tai001_20_5	1284,28	1289,58	6,95	1293,32	1297,60	6,81
tai002_20_5	1361,07	1368,07	6,93	1365,09	1370,50	6,90
tai003_20_5	1090,80	1093,79	6,89	1108,25	1108,90	6,74

Continua na página seguinte

Tabela 19 – Resultados obtidos pelos métodos QLNEH-S e IQL-S (Continuação)

Instância	**QLNEH-S**			**IQL-S**		
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**
tai004_20_5	1326,29	1323,33	7,23	1320,47	1321,60	7,13
tai005_20_5	1259,61	1255,01	7,53	1248,83	1250,94	7,39
tai006_20_5	1211,72	1216,92	7,47	1210,63	1214,73	7,44
tai007_20_5	1249,03	1253,95	7,49	1254,01	1256,20	7,42
tai008_20_5	1222,25	1226,85	7,51	1225,17	1229,51	7,35
tai009_20_5	1254,40	1259,76	7,60	1257,97	1260,23	7,44
tai010_20_5	1127,54	1129,64	7,54	1130,58	1126,29	7,41
tai011_20_10	1634,81	1638,25	12,56	1616,56	1614,85	12,43
tai012_20_10	1715,45	1717,29	12,43	1711,73	1708,46	12,27
tai013_20_10	1546,91	1543,01	12,43	1556,19	1565,05	12,21
tai014_20_10	1419,47	1420,82	12,61	1411,56	1414,55	12,22
tai015_20_10	1456,13	1459,04	12,45	1464,34	1462,68	12,33
tai016_20_10	1438,94	1442,94	12,40	1433,38	1436,66	12,34
tai017_20_10	1517,43	1518,97	12,40	1519,95	1519,75	12,31
tai018_20_10	1586,77	1588,76	12,40	1586,88	1588,23	12,26
tai019_20_10	1639,56	1640,20	12,43	1642,76	1641,58	12,27
tai020_20_10	1649,40	1647,72	12,53	1649,75	1652,50	12,31
tai021_20_20	2342,57	2343,49	22,34	2353,88	2355,32	22,00
tai022_20_20	2149,21	2141,86	22,76	2147,58	2142,62	21,90
tai023_20_20	2394,72	2404,58	22,36	2402,96	2401,52	21,98
tai024_20_20	2272,37	2280,37	22,04	2274,33	2266,27	21,92
tai025_20_20	2367,78	2373,29	22,03	2354,21	2351,01	21,95
tai026_20_20	2302,40	2301,86	22,35	2294,68	2296,86	22,06
tai027_20_20	2363,22	2372,77	22,01	2336,72	2336,73	22,06
tai028_20_20	2248,44	2248,04	22,12	2257,45	2259,35	21,99
tai029_20_20	2294,31	2295,02	21,98	2288,37	2296,41	21,88
tai030_20_20	2252,95	2254,59	22,20	2248,12	2250,12	21,90
tai031_50_5	2728,84	2738,40	44,40	2731,19	2740,70	42,80

Continua na página seguinte

Tabela 19 – Resultados obtidos pelos métodos QLNEH-S e IQL-S (Continuação)

Instância	**QLNEH-S**			**IQL-S**		
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**
tai032_50_5	2869,21	2878,31	44,33	2875,42	2874,61	42,82
tai033_50_5	2644,78	2647,79	44,27	2651,96	2649,56	42,58
tai034_50_5	2782,49	2787,61	44,88	2780,42	2783,97	43,34
tai035_50_5	2878,56	2883,55	44,71	2881,29	2881,46	43,30
tai036_50_5	2846,19	2847,38	44,65	2840,96	2847,29	43,12
tai037_50_5	2770,83	2776,74	44,79	2766,64	2767,76	43,09
tai038_50_5	2723,01	2727,17	44,30	2714,94	2723,50	42,86
tai039_50_5	2595,15	2594,17	44,30	2588,88	2588,29	42,78
tai040_50_5	2790,60	2798,74	44,37	2784,36	2786,37	43,21
tai041_50_10	3161,18	3158,65	72,03	3157,00	3159,85	69,99
tai042_50_10	3025,45	3025,08	72,47	3047,41	3041,15	69,69
tai043_50_10	3015,49	3020,50	71,60	3041,97	3040,41	70,15
tai044_50_10	3180,80	3175,92	72,07	3149,84	3150,70	69,63
tai045_50_10	3142,65	3147,61	73,63	3154,44	3156,65	70,31
tai046_50_10	3156,63	3159,65	73,65	3156,62	3153,85	74,43
tai047_50_10	3231,03	3233,60	73,59	3248,87	3238,06	70,92
tai048_50_10	3139,96	3149,33	73,72	3176,98	3173,37	69,55
tai049_50_10	3052,15	3053,29	73,86	3044,29	3046,94	69,70
tai050_50_10	3223,91	3220,30	73,19	3249,26	3246,11	69,35
tai051_50_20	4082,33	4087,91	129,35	4111,16	4120,88	123,06
tai052_50_20	3969,09	3969,42	129,09	3983,28	3980,03	122,81
tai053_50_20	3891,25	3887,57	128,74	3886,41	3893,09	123,31
tai054_50_20	3937,85	3944,20	128,07	3972,19	3979,25	124,27
tai055_50_20	3884,32	3882,40	128,29	3872,97	3864,46	123,04
tai056_50_20	3930,24	3936,66	126,24	3906,46	3910,36	123,03
tai057_50_20	3989,91	3989,03	125,93	3920,12	3926,08	123,33
tai058_50_20	3957,71	3962,08	125,62	3941,28	3945,74	123,18
tai059_50_20	3978,20	3975,68	126,10	3995,39	3998,85	123,45

Continua na página seguinte

Tabela 19 – Resultados obtidos pelos métodos QLNEH-S e IQL-S (Continuação)

Instância	**QLNEH-S**			**IQL-S**		
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**
tai060_50_20	3969,26	3967,80	127,23	3977,23	3974,85	122,61
tai061_100_5	5505,29	5520,74	235,15	5510,73	5530,04	218,54
tai062_100_5	5298,60	5309,30	235,78	5274,32	5288,57	220,59
tai063_100_5	5234,32	5246,37	235,97	5215,22	5214,31	220,57
tai064_100_5	5042,40	5051,22	234,73	5041,59	5040,65	218,14
tai065_100_5	5276,78	5278,30	240,38	5285,28	5296,50	220,69
tai066_100_5	5148,51	5152,66	236,13	5148,75	5153,15	219,95
tai067_100_5	5279,28	5295,91	236,59	5276,77	5280,85	220,21
tai068_100_5	5131,39	5144,67	237,23	5130,04	5142,27	221,71
tai069_100_5	5503,00	5509,89	238,04	5487,61	5496,41	221,53
tai070_100_5	5357,51	5352,25	236,57	5343,83	5349,45	220,45
tai071_100_10	5921,54	5937,36	389,56	5919,87	5934,05	375,72
tai072_100_10	5499,78	5510,21	396,69	5492,23	5491,25	375,45
tai073_100_10	5802,30	5804,86	403,28	5769,28	5769,78	376,17
tai074_100_10	6030,15	6044,55	397,75	6028,86	6035,26	380,71
tai075_100_10	5676,94	5671,93	397,04	5679,38	5680,79	371,99
tai076_100_10	5421,24	5421,97	398,65	5434,62	5432,20	376,56
tai077_100_10	5728,43	5734,07	402,52	5743,64	5736,06	378,43
tai078_100_10	5813,24	5816,48	399,15	5817,26	5818,81	375,67
tai079_100_10	6008,51	6013,29	399,35	6007,35	6001,10	374,47
tai080_100_10	5928,73	5930,27	393,27	5959,35	5954,42	380,25
tai081_100_20	6650,47	6653,11	709,12	6662,80	6658,22	681,76
tai082_100_20	6574,55	6572,84	702,93	6632,90	6631,86	680,97
tai083_100_20	6682,31	6691,60	718,82	6686,17	6686,22	676,88
tai084_100_20	6641,77	6658,38	706,76	6648,69	6654,69	682,15
tai085_100_20	6710,05	6709,90	710,95	6698,06	6695,73	691,09
tai086_100_20	6780,20	6782,19	707,10	6773,52	6778,31	674,08
tai087_100_20	6705,49	6697,93	705,40	6725,39	6737,24	676,90

Continua na página seguinte

Tabela 19 – Resultados obtidos pelos métodos QLNEH-S e IQL-S (Continuação)

Instância	**QLNEH-S**			**IQL-S**		
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**
tai088_100_20	6879,20	6880,36	696,66	6863,35	6869,48	674,07
tai089_100_20	6708,74	6707,17	700,84	6726,91	6738,14	673,69
tai090_100_20	6816,46	6822,95	694,23	6819,57	6821,68	672,08
tai091_200_10	11046,41	11043,58	2759,66	11009,54	11024,41	2515,83
tai092_200_10	10803,45	10795,13	2773,65	10780,08	10783,75	2495,92
tai093_200_10	11189,55	11193,41	2775,05	11173,51	11179,82	2506,85
tai094_200_10	11025,71	11023,68	2760,20	11005,18	11004,22	2502,33
tai095_200_10	10763,03	10756,77	2788,24	10787,31	10796,57	2484,42
tai096_200_10	10574,63	10586,56	2780,52	10556,81	10570,66	2482,73
tai097_200_10	11086,13	11087,58	2765,01	11034,73	11038,31	2521,37
tai098_200_10	10915,53	10924,11	2780,18	10970,41	10973,68	2490,66
tai099_200_10	10699,91	10699,67	2761,33	10686,44	10690,96	2483,68
tai100_200_10	10875,29	10870,90	2767,73	10899,76	10902,81	2476,67
tai101_200_20	11851,76	11849,16	4944,48	11815,54	11813,69	4650,78
tai102_200_20	11975,38	11974,74	4958,70	11982,87	11979,55	4626,47
tai103_200_20	12025,74	12032,75	4975,92	12025,59	12033,56	4637,26
tai104_200_20	11942,95	11934,54	5017,22	11957,08	11971,45	4619,78
tai105_200_20	11881,26	11875,61	5003,67	11906,19	11914,99	4584,93
tai106_200_20	11849,42	11850,82	5077,50	11881,20	11887,24	4614,98
tai107_200_20	12082,67	12087,90	5136,07	12082,95	12076,16	4626,71
tai108_200_20	12028,80	12044,18	5002,63	12038,08	12037,33	4581,23
tai109_200_20	11868,71	11874,88	4893,46	11932,45	11936,13	4661,29
tai110_200_20	12018,48	12026,56	4911,34	12027,61	12052,79	4646,53

Fonte: Elaborada pelo autor (2025)

Tabela 20 – Resultados obtidos pelos métodos NEHTBLMED e GRASPMED

Instância	NEHTBLMED			GRASPMED			
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**	**Nº It.**
tai001_20_5	1286	1291,41	0,02	1356	1451,34	0,11	10
tai002_20_5	1365	1368,84	0,01	1450	1553,46	0,11	10
tai003_20_5	1132	1144,65	0,01	1327	1455,47	0,11	10
tai004_20_5	1329	1341,59	0,01	1511	1632,72	0,11	10
tai005_20_5	1277	1305,60	0,01	1360	1456,08	0,11	10
tai006_20_5	1224	1234,09	0,01	1401	1527,90	0,11	10
tai007_20_5	1270	1282,38	0,01	1343	1423,91	0,11	10
tai008_20_5	1223	1232,50	0,01	1417	1559,35	0,11	10
tai009_20_5	1285	1297,69	0,01	1506	1607,47	0,11	10
tai010_20_5	1151	1158,18	0,01	1293	1447,44	0,11	10
tai011_20_10	1675	1682,00	0,02	1826	1947,75	0,21	10
tai012_20_10	1723	1735,03	0,02	1919	2005,07	0,21	10
tai013_20_10	1538	1559,22	0,02	1745	1986,17	0,21	10
tai014_20_10	1429	1444,61	0,03	1683	1815,25	0,23	10
tai015_20_10	1493	1507,18	0,03	1717	1839,54	0,23	10
tai016_20_10	1441	1457,95	0,03	1736	1842,57	0,23	10
tai017_20_10	1539	1562,94	0,03	1690	1851,20	0,23	10
tai018_20_10	1605	1614,84	0,03	1803	1926,41	0,23	10
tai019_20_10	1645	1651,12	0,03	1796	1916,36	0,23	10
tai020_20_10	1649	1662,07	0,03	1874	2016,94	0,23	10
tai021_20_20	2407	2420,94	0,05	2593	2745,02	0,46	10
tai022_20_20	2150	2158,88	0,05	2460	2582,93	0,46	10
tai023_20_20	2429	2440,21	0,05	2628	2762,82	0,46	10
tai024_20_20	2251	2271,15	0,05	2479	2659,44	0,49	10
tai025_20_20	2392	2401,86	0,05	2573	2709,51	0,46	10
tai026_20_20	2322	2358,09	0,05	2515	2630,54	0,46	10
tai027_20_20	2361	2370,87	0,05	2566	2766,19	0,46	10
tai028_20_20	2239	2261,05	0,05	2529	2716,90	0,46	10

Continua na página seguinte

Tabela 20 – Resultados obtidos pelos métodos NEHTBLMED e GRASPMED (Continuação)

Instância	**NEHTBLMED**			**GRASPMED**			
	MM	**MMC**	**T(s)**	**MM**	**MMC**	**T(s)**	**Nº It.**
tai029_20_20	2306	2316,63	0,05	2553	2829,57	0,46	10
tai030_20_20	2277	2285,45	0,05	2497	2705,73	0,46	10
tai031_50_5	2729	2740,57	0,18	3058	3206,96	1,73	10
tai032_50_5	2882	2883,52	0,18	3287	3402,78	1,75	10
tai033_50_5	2625	2637,71	0,18	3115	3255,47	1,76	10
tai034_50_5	2782	2792,30	0,18	3199	3368,34	1,77	10
tai035_50_5	2864	2874,42	0,18	3243	3390,65	1,75	10
tai036_50_5	2835	2848,01	0,18	3206	3329,38	1,76	10
tai037_50_5	2746	2765,77	0,18	3098	3224,24	1,76	10
tai038_50_5	2703	2716,63	0,18	3086	3249,86	1,74	10
tai039_50_5	2569	2582,73	0,18	2811	2929,90	1,74	10
tai040_50_5	2789	2800,02	0,18	3222	3397,88	1,74	10
tai041_50_10	3133	3154,05	0,37	3691	3849,39	3,55	10
tai042_50_10	3022	3049,57	0,37	3485	3663,75	3,57	10
tai043_50_10	2993	3009,26	0,37	3586	3720,12	3,59	10
tai044_50_10	3167	3191,50	0,37	3739	3890,42	3,53	10
tai045_50_10	3179	3204,11	0,37	3845	3985,98	3,55	10
tai046_50_10	3158	3182,44	0,37	3771	3915,65	3,58	10
tai047_50_10	3241	3290,12	0,37	3735	3891,14	3,52	10
tai048_50_10	3149	3171,03	0,37	3634	3790,08	3,53	10
tai049_50_10	2996	3012,86	0,37	3643	3785,85	3,53	10
tai050_50_10	3250	3277,01	0,37	3739	3870,14	3,53	10
tai051_50_20	4026	4055,55	0,74	4774	4896,20	7,09	10
tai052_50_20	3906	3936,03	0,74	4520	4680,48	7,13	10
tai053_50_20	3916	3943,22	0,74	4503	4631,56	7,09	10
tai054_50_20	3959	4007,09	0,74	4508	4598,07	7,22	10
tai055_50_20	3806	3837,73	0,74	4701	4886,51	7,14	10
tai056_50_20	3904	3936,34	0,74	4701	4809,40	7,11	10

Continua na página seguinte

Tabela 20 – Resultados obtidos pelos métodos NEHTBLMED e GRASPMED (Continuação)

Instância	NEHTBLMED			GRASPMED			
	MM	MMC	T(s)	MM	MMC	T(s)	Nº It.
tai057_50_20	3941	3972,85	0,74	4710	4889,23	7,13	10
tai058_50_20	3925	3948,62	0,74	4661	4803,54	7,05	10
tai059_50_20	3932	3973,35	0,74	4587	4711,97	7,07	10
tai060_50_20	3993	4036,37	0,74	4698	4852,24	7,10	10
tai061_100_5	5519	5538,14	1,42	6457	6649,42	13,86	10
tai062_100_5	5329	5360,05	1,42	6248	6415,08	13,85	10
tai063_100_5	5206	5230,58	1,40	6116	6269,75	13,75	10
tai064_100_5	5023	5042,84	1,41	5682	5846,39	13,91	10
tai065_100_5	5255	5289,09	1,41	6121	6269,88	13,82	10
tai066_100_5	5127	5144,94	1,42	6060	6201,50	13,83	10
tai067_100_5	5219	5252,73	1,41	6030	6162,76	13,86	10
tai068_100_5	5289	5309,88	1,41	6126	6265,56	13,82	10
tai069_100_5	5125	5162,77	1,42	6043	6194,45	13,85	10
tai070_100_5	5046	5067,88	1,42	5677	5825,61	13,91	10
tai071_100_10	6462	6509,09	2,81	7584	7849,93	27,90	10
tai072_100_10	6408	6449,84	2,81	7524	7762,23	27,90	10
tai073_100_10	6154	6196,87	2,81	7267	7504,42	27,87	10
tai074_100_10	6256	6284,52	2,81	7391	7625,47	27,92	10
tai075_100_10	6308	6335,80	2,81	7406	7645,02	27,90	10
tai076_100_10	6374	6410,27	2,81	7480	7716,07	27,89	10
tai077_100_10	6298	6334,80	2,81	7384	7619,40	27,89	10
tai078_100_10	6138	6173,86	2,81	7209	7434,38	27,88	10
tai079_100_10	6355	6388,85	2,81	7443	7671,23	27,88	10
tai080_100_10	6175	6217,13	2,81	7237	7461,13	27,89	10
tai081_100_20	6553	6599,22	5,74	7777	7939,70	56,32	10
tai082_100_20	6462	6497,47	5,78	7728	7894,67	56,35	10
tai083_100_20	6557	6597,19	5,73	7943	8112,06	56,38	10
tai084_100_20	6555	6610,20	5,73	8001	8154,61	56,87	10

Continua na página seguinte

Tabela 20 – Resultados obtidos pelos métodos NEHTBLMED e GRASPMED (Continuação)

Instância	NEHTBLMED			GRASPMED			
	MM	MMC	T(s)	MM	MMC	T(s)	Nº It.
tai085_100_20	6635	6689,80	5,73	7980	8176,83	56,56	10
tai086_100_20	6654	6689,78	5,72	7956	8152,50	56,86	10
tai087_100_20	6587	6643,89	5,76	7970	8132,28	57,94	10
tai088_100_20	6827	6874,50	5,90	7883	8029,33	56,51	10
tai089_100_20	6661	6701,36	5,73	7841	8057,39	56,58	10
tai090_100_20	6666	6698,30	5,73	7873	8054,43	57,25	10
tai091_200_10	10965	10997,08	22,35	12691	12881,72	224,55	10
tai092_200_10	10649	10701,18	22,45	12919	13097,22	223,82	10
tai093_200_10	11069	11108,73	22,19	12968	13140,05	222,24	10
tai094_200_10	11019	11060,04	22,31	12814	12986,70	222,86	10
tai095_200_10	10682	10730,14	22,53	12783	13007,88	221,28	10
tai096_200_10	10451	10491,17	22,86	12780	12985,24	225,63	10
tai097_200_10	10941	10980,98	22,50	13207	13383,59	222,11	10
tai098_200_10	10837	10877,11	22,50	13346	13538,98	222,12	10
tai099_200_10	10617	10671,97	22,45	12722	12892,20	224,00	10
tai100_200_10	10813	10879,09	22,61	13000	13204,06	225,12	10
tai101_200_20	11666	11731,70	45,46	14076	14236,17	452,75	10
tai102_200_20	11715	11782,80	45,33	14149	14325,99	454,57	10
tai103_200_20	11919	11979,31	45,54	14285	14462,78	452,29	10
tai104_200_20	11777	11834,54	45,44	13690	13866,13	451,40	10
tai105_200_20	11582	11636,50	45,26	13996	14226,93	449,36	10
tai106_200_20	11648	11710,69	45,32	14131	14332,44	449,43	10
tai107_200_20	11836	11893,67	45,35	14145	14331,38	484,81	10
tai108_200_20	11825	11890,89	45,67	14328	14513,59	478,30	10
tai109_200_20	11640	11706,65	45,58	14085	14259,12	448,68	10
tai110_200_20	11805	11881,63	45,26	14132	14328,74	452,94	10

Fonte: Elaborada pelo autor (2025)