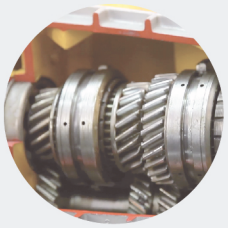
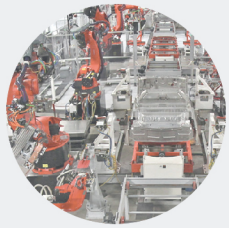
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA E GESTÃO INDUSTRIAL



Sistema de Seleção e Autoparametrização de Meta-heurísticas com Machine Learning

ANA BEATRIZ DA COSTA SILVA
junho de 2025

POLITÉCNICO
DO PORTO

ISEP INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

Sistema de Seleção e Autoparametrização de Meta-heurísticas com Machine Learning

Ana Beatriz da Costa Silva

Dissertação para obtenção do Grau de Mestre em Engenharia e Gestão Industrial

Orientador: Ivo André Soares Pereira

Júri:

Presidente:

Prof. Dr.ª Susana Nicola, Professora Adjunta, Instituto Superior de Engenharia do Porto

Vogais:

Prof. Dr. Bruno Cunha, Professor Auxiliar, Universidade Portucalense

Prof. Dr. Ivo Pereira, Professor Adjunto, Instituto Superior de Engenharia do Porto

Porto - junho, 2025

Agradecimentos

Aproveito esta oportunidade para agradecer a todas as pessoas que, de alguma forma, deram o seu contributo para que a elaboração deste estudo fosse possível.

Para além de todo o apoio sentido, quer da parte dos professores e colegas, agradeço especialmente ao meu orientador, professor Ivo Pereira, por toda a ajuda, interesse, disponibilidade, concelhos e críticas construtivas que levaram à concretização deste projeto.

À minha família, em especial aos meus pais e ao meu irmão, por todo o carinho, apoio e motivação que me ofereceram principalmente nesta etapa.

A todos, o meu sincero obrigado!

Resumo

O escalonamento de produção apresenta como objetivo principal a definição de um plano que atinja as metas de produção, garantindo que todas as restrições sejam cumpridas e otimizadas. Consiste num processo que desempenha um papel fundamental nas empresas, identificando-se como um fator que influencia a eficiência operacional e qualidades das organizações. A resolução de problemas de escalonamento é uma tarefa complexa e necessita da implementação de metodologias adequadas. A principal motivação deste trabalho surgiu da necessidade de oferecer uma contribuição para a resolução de problemas de escalonamento.

As Meta-heurísticas, caraterizadas pela sua fácil implementação, podem ser utilizadas para a resolução de problemas de escalonamento. No entanto, a sua aplicação apresenta desafios, principalmente na escolha da técnica mais adequada para a resolução de um determinado problema. Além disso, as Meta-heurísticas possuem parâmetros que necessitam de ser definidos de forma adequada.

As Meta-heurísticas e a Machine Learning são frequentemente combinadas com o intuito de diminuir as suas desvantagens, melhor as suas capacidades e, com isso, melhorar o desempenho do sistema de produção. As abordagens de Machine Learning podem ser integradas com o intuito de auxiliar na seleção e parametrização das Meta-heurísticas.

Neste contexto, surge a necessidade de desenvolver um sistema de seleção e autoparametrização de Meta-heurísticas que utiliza Machine Learning para otimizar o escalonamento de tarefas em ambientes de produção. Este sistema abrange unicamente problemas de escalonamento do tipo Job-Shop e a parametrização das Meta-heurísticas é realizada de forma *offline*. O módulo considera apenas técnicas de aprendizagem supervisionada de Machine Learning.

O sistema proposto foi estruturado em dois modelos preditivos: o primeiro destina-se à seleção da Meta-heurística mais adequada para um problema de escalonamento; o segundo é responsável pela afinação dos parâmetros da Meta-heurística selecionada. Os dois modelos preditivos foram avaliados com recurso a indicadores de desempenho adequados, permitindo a realização de uma análise detalhada da eficácia da integração das técnicas de Machine Learning.

Palavras-chave: Escalonamento; Meta-heurísticas; Machine Learning; Afinação; Parâmetros

Abstract

The main objective of production scheduling is to define a plan that achieves production targets, ensuring that all constraints are met and optimised. It is a process that plays a fundamental role in companies, identifying itself as a factor that influences the operational efficiency and quality of organisations. Solving scheduling problems is a complex task and requires the implementation of appropriate methodologies. The main motivation for this work arose from the need to contribute to the resolution of scheduling problems.

Meta-heuristics, characterised by their easy implementation, can be used to solve scheduling problems. However, their application presents challenges, mainly in choosing the most appropriate technique for solving a given problem. In addition, Meta-heuristics have parameters that need to be defined appropriately.

Meta-heuristics and Machine Learning are often combined to reduce their disadvantages, improve their capabilities and, thereby, improve the performance of the production system. Machine learning approaches can be integrated to assist in the selection and parameterisation of meta-heuristics.

In this context, there is a need to develop a Meta-heuristic selection and self-parameterisation system that uses Machine Learning to optimise task scheduling in production environments. This system only covers Job-Shop scheduling problems, and the parameterisation of Meta-heuristics is performed offline. The module only considers supervised Machine Learning techniques.

The proposed system was structured into two predictive models: the first is intended for selecting the most appropriate Meta-heuristic for a scheduling problem; the second is responsible for fine-tuning the parameters of the selected Meta-heuristic. The two predictive models were evaluated using appropriate performance indicators, allowing for a detailed analysis of the effectiveness of integrating Machine Learning techniques.

Keywords: Scheduling; Meta-heuristics; Machine Learning; Tuning; Parameters

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

ACO	Ant Colony Optimization
AUC	Area Under the Curve
CRISP-DM	Cross Industry Standard Process for Data Mining
DOE	Design of Experiments
GA	Algoritmos Genéticos
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MILP	Programação Linear Inteira Mista
MSE	Mean Squared Error
PSO	Particle Swarm Optimization
RMSE	Root Mean Squared Error
ROC	Receiver Operating Characteristic
SA	Simulated Annealing
TS	Tabu Search
WT	Weighted Tardiness

Lista de Símbolos

R^2	Coeficiente de Determinação

1 Introdução

Este capítulo descreve os aspetos que incentivaram o desenvolvimento desta dissertação, sendo apresentado uma pequena explicação do contexto em que se insere o trabalho abordado. As questões de investigação são enunciadas, seguidas do objetivo global e dos objetivos específicos que se pretendem alcançar ao longo do desenvolvimento do projeto. Adicionalmente, são apresentadas as opções metodológicas adotadas no trabalho.

1.1 Problema de investigação, enquadramento e pertinência

O padrão de exigência do mercado encontra-se em constante crescimento, impulsionado pela globalização, a transformação digital e as expectativas cada vez mais elevadas dos consumidores. Para as empresas, isto implica a necessidade de responder a critérios rigorosos de qualidade, eficiência e inovação, a fim de se manterem competitivas e pertinentes.

Em ambientes reais de produção, o escalonamento, procedimento de alocação de tarefas às máquinas, é um fator que influência a eficiência operacional da empresa. Organizações que realizam uma boa distribuição de atividades pelas máquinas são diferenciadas pelos seus altos níveis de qualidade e desempenho (Wang, Pan and Wang, 2021).

O escalonamento de produção é um método de tomada de decisões que desempenha um papel crucial nos sistemas de fabrico. O objetivo principal é atribuir os recursos de produção disponíveis, como as máquinas, às atividades e decidir a sequência de tarefas de modo que todas as restrições sejam cumpridas e os objetivos otimizados (Lei and Cai, 2020).

O escalonamento de produção tem sido amplamente estudado ao longo dos anos, atraindo a atenção de investigadores especializados em gestão, engenharia industrial, investigação operacional e computação. Um problema de escalonamento de produção, normalmente, assume um número fixo de tarefas, onde cada tarefa possui os seus próprios parâmetros

específicos, isto é, restrições de sequência, estimativas de tempo de cada operação e os recursos necessários (Fuchigami and Rangel, 2018). Devido à sua natureza dinâmica e alta complexidade de resolução, os problemas de escalonamento são considerados como NP-difíceis (Wang, Pan and Wang, 2021).

A resolução deste tipo de problemas exige a aplicação de algoritmos adequados, que podem ser divididos em duas categorias: métodos exatos e métodos por aproximação. Os métodos exatos garantem a obtenção de uma solução ótima global, sendo especialmente usados para resolver problemas que exijam um elevado nível precisão (Xu *et al.*, 2022). Dentro dos diversos métodos exatos, a Programação Linear Inteira Mista (MILP) sobressai pela sua ampla utilização na resolução de problemas de otimização. Nos últimos anos, o desenvolvimento de software e as melhorias na capacidade de processamento, tornaram a aplicação de Programação Linear Inteira Mista mais prática, acessível e eficiente (Klein, Gnägi and Trautmann, 2024). No entanto, os métodos exatos não podem ser utilizados em problemas complexos. Estas técnicas possuem uma capacidade limitada, sendo aplicáveis apenas a problemas de pequena escala. Para além disto, apresentam um desempenho lento (Xu *et al.*, 2022).

Os métodos por aproximação são adequados para a resolução de problemas de grande escala, atendendo de forma mais eficaz às necessidades dos desafios do mundo real de produção. Tratam-se de abordagens que não garantem a aquisição de soluções ótimas, mas, proporcionam soluções rápidas. Os métodos por aproximação são compostos pelos Métodos de Interpolação, Redes Neuronais, algoritmos de Pesquisa Local, Meta-heurísticas, entre outros (Xu *et al.*, 2022). Nesta dissertação, destaca-se a aplicação de Meta-heurísticas que fornecem soluções satisfatórias em tempos de execução razoáveis.

As Meta-heurísticas podem ser facilmente modificadas de acordo com o problema em causa, caraterizando-se por serem técnicas de otimização capazes de evitar ótimos locais e apresentam a vantagem de fácil implementação, simplicidade e flexibilidade (Agrawal *et al.*, 2021). No entanto, as Meta-heurísticas apresentam alguns desafios, particularmente a identificação da Meta-heurística mais adequada para a resolução de um problema é uma etapa complexa, visto que é difícil eleger uma Meta-heurística como a melhor de todas. Cada Meta-heurística possui vantagens e desvantagens e a sua eficácia depende das propriedades do problema em análise. Adicionalmente, as Meta-heurísticas possuem parâmetros que, para alcançar um bom desempenho, necessitam de ser definidos de forma apropriada (Calvet *et al.*, 2017; Pereira *et al.*, 2021).

As Meta-heurísticas e várias técnicas de Machine Learning têm sido estudadas e registaram avanços significativos nos seus respetivos domínios, sendo frequentemente combinadas para minimizar as suas desvantagens e melhorar as suas capacidades. Particularmente, a incorporação de Machine Learning na parametrização das Meta-heurísticas pode incrementar o desempenho, aumentando os resultados do sistema de produção. A Machine Learning é uma abordagem que dá às máquinas a capacidade de aprender automaticamente com dados e experiências, identificando padrões para realizar previsões com o mínimo de intervenção humana. Nos últimos anos, houve um aumento de estudos relacionados com esta combinação

de metodologias, indicando um enorme interesse para a resolução de problemas complexos nas áreas da engenharia (Talbi, 2021; Cuevas, Zaldívar and Pérez, 2022; Karimi-Mamaghan *et al.*, 2022).

A principal motivação deste trabalho surgiu da necessidade de desenvolver um sistema de seleção e autoparametrização de Meta-heurísticas utilizando Machine Learning para otimizar o escalonamento de tarefas em ambientes de produção. Com o aumento da variabilidade e complexidade dos processos produtivos, a eficiência no escalonamento torna-se crucial para melhorar a produtividade. Deste modo, através da exploração de algoritmos de aprendizagem supervisionada, pretende-se melhorar a eficiência dos processos de escalonamento ao identificar e ajustar dinamicamente os parâmetros das Meta-heurísticas.

1.2 Questões e objetivos de investigação

A implementação de algoritmos de Meta-heurísticas tem-se revelado como uma abordagem inovadora para enfrentar os desafios de escalonamento, destacando a importância da definição precisa dos parâmetros para assegurar a eficiência do sistema de produção. A integração de Machine Learning permite a obtenção de informações relevantes, possibilitando a identificação de padrões e a previsão dos parâmetros mais adequados, aumentando o desempenho das Meta-heurísticas (Karimi-Mamaghan *et al.*, 2022; Xu *et al.*, 2022).

Neste âmbito, surgem as seguintes questões de investigação:

- De que modo os algoritmos de Machine Learning contribuem para prever os melhores parâmetros para diferentes instâncias de problemas de escalonamento?
- Quais são os principais desafios e limitações na implementação de um sistema de seleção e autoparametrização de Meta-heurísticas em problemas de escalonamento?

Nesta dissertação, o objetivo geral é desenvolver um sistema de seleção e autoparametrização de Meta-heurísticas, que utiliza Machine Learning para identificar e ajustar os parâmetros da Meta-heurística mais adequada para a resolução de um determinado problema. Para tal, serão abordados os seguintes pontos para a realização de uma análise aprofundada:

- Analisar as principais técnicas de Meta-heurísticas e identificar os parâmetros que influenciam o seu desempenho.
- Avaliar e selecionar técnicas de Machine Learning adequadas para seleção e para a parametrização das Meta-heurísticas.
- Implementar modelos preditivos para selecionar a Meta-heurística mais adequada para a resolução de um problema de escalonamento e afinar os seus parâmetros.

1.3 Opções metodológicas

A dissertação apresentada neste documento adota uma abordagem dedutiva para o desenvolvimento do estudo. Esta metodologia baseia-se na aplicação de princípios teóricos

estabelecidos na literatura sobre Meta-heurísticas e Machine Learning para resolver problemas de escalonamento de produção.

Na elaboração do projeto, será aplicado o modelo CRISP-DM, Cross Industry Standard Process for Data Mining. O CRISP-DM é um modelo composto por seis fases iterativas que descreve o ciclo de vida da ciência dos dados. (Schröer, Kruse and Gómez, 2021).

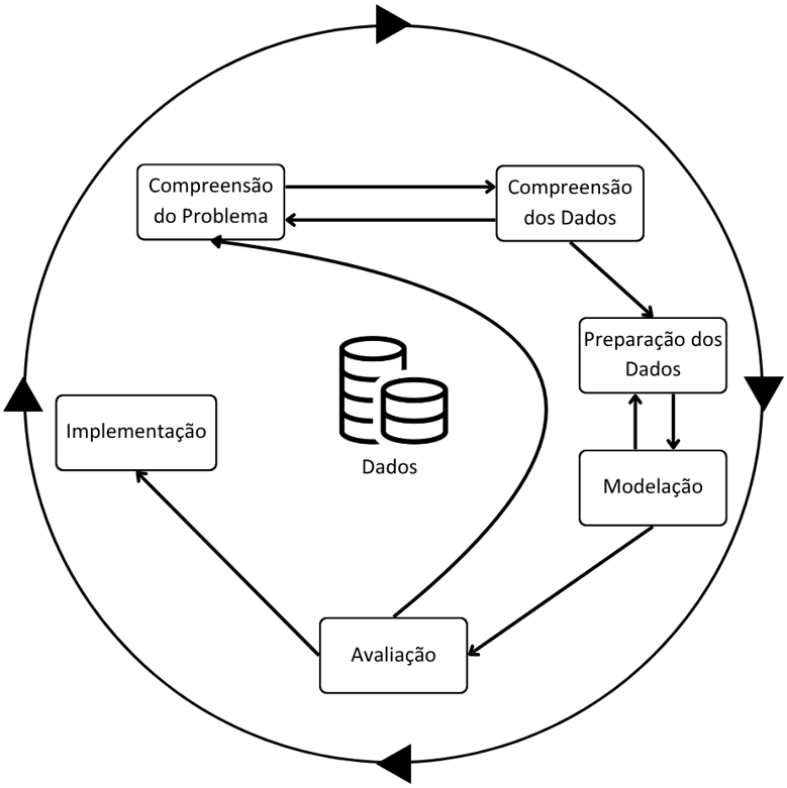


Figura 1 - Metodologia CRISP-DM, baseado em Schröer, Kruse and Gómez (2021)

As etapas do modelo CRISP-DM, considerando o tema da dissertação, são:

- **Compreensão do Problema:** Definir o contexto de escalonamento de produção e identificar as Meta-heurísticas e as técnicas de Machine Learning adotadas no estudo.
- **Compreensão dos Dados:** Recolher dados sobre o desempenho das Meta-heurísticas em diferentes instâncias de escalonamento e realizar uma análise exploratória com o intuito de identificar padrões relevantes nos dados.
- **Preparação dos Dados:** Organização dos dados brutos em um formato adequado para a etapa da Modelação.
- **Modelação:** Elaborar modelos preditivos que utilizem algoritmos de Machine Learning para identificar e prever os melhores parâmetros da Meta-heurística.
- **Avaliação:** Validar a eficácia dos modelos utilizando métricas de desempenho adequadas.
- **Implementação:** Desenvolver uma proposta de um sistema de seleção e autoparametrização das Meta-heurísticas que permite a aplicação de novas instâncias de problemas de escalonamento de tempo real.

1.4 Estrutura do documento

Esta dissertação é composta por seis capítulos. Após o capítulo de introdução, o capítulo 2 é dedicado à revisão bibliográfica, onde são abordados diversos temas relevantes para o desenvolvimento deste projeto. No capítulo 2, inicialmente, é apresentado o conceito de escalonamento de produção, com uma caraterização das duas técnicas de escalonamento mais estudadas, o Flow-Shop e o Job-Shop. Em seguida, é explorada a metodologia das Meta-heurísticas, com a descrição de alguns algoritmos, nomeadamente: Simulated Annealing, Tabu Search, Algoritmos Genéticos, Ant Colony Optimization e Artifical Bee Colony. É também elaborada uma análise sobre o problema de escalonamento de parâmetros.

Adicionalmente, é abordada a metodologia de Machine Learning, com a descrição das diferentes classificações desta abordagem: aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço. São também destacados alguns algoritmos de aprendizagem supervisionada, nomeadamente: Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest, Regressão Linear e Regressão Logística. No capítulo 2, por fim, é apresentada uma revisão de estudos relacionados com esta dissertação, com destaque para os projetos que abordam a seleção dos algoritmos para resolução de problemas de escalonamento e a afinação de parâmetros das Meta-heurísticas.

O capítulo 3 apresenta uma descrição de como foi implementado o CRISP-DM nesta dissertação. Além disso, é realizada uma identificação e contextualização do problema que incentiva a elaboração desta dissertação, juntamente com uma análise dos dados brutos utilizados neste projeto.

Os capítulos 4 e 5 são dedicados à apresentação das etapas preparação de dados, modelação e avaliação correspondentes à elaboração dos modelos preditivos de seleção da Meta-heurística e afinação dos parâmetros da Meta-heurística, respetivamente. O capítulo 6 corresponde ao último capítulo onde são apresentadas as conclusões deste projeto.

2 Revisão bibliográfica

Este capítulo tem como objetivo fornecer uma análise detalhada dos principais conceitos e metodologias utilizadas no desenvolvimento desta dissertação. Em primeiro lugar, é apresentado o problema de escalonamento, juntamente com os dois tipos de ambiente de produção mais estudados, o Flow-Shop e o Job-Shop. Em seguida, explora-se a metodologia de Meta-Heurística, abordando o desafio de afinação de parâmetros. Depois é introduzida a temática de Machine Learning destacando alguns algoritmos de aprendizagem supervisionada. Por último, são apresentados alguns estudos relacionados com o tema desta dissertação.

2.1 Problema de escalonamento

O escalonamento de produção representa uma atividade crítica no âmbito operacional das empresas, garantindo a sua elevada competitividade nos mercados atuais. Trata-se de um processo de tomada de decisão que aborda a afetação de recursos, como seres humanos e máquinas, a tarefas, numa sequência específica e em períodos previamente definidos (Coelho and Silva, 2021). O principal objetivo dos problemas de escalonamento é apresentar um plano que atinja as metas de produção, respeitando todas as restrições operacionais e técnicas, e que alcance um ou mais objetivos (Georgiadis, Elekidis and Georgiadis, 2019).

Os problemas de escalonamento são caraterizados por uma variedade de critérios. Entre esses critérios, destaca-se a **função objetivo**, que pode estar associada a alguns parâmetros, tal como o tempo, custo ou número de tarefas. O **ambiente de produção** é outro critério relevante, sendo definido pela quantidade e tipo de recursos utilizados e pela forma como estão dispostos para realizar as etapas do processo produtivo. Adicionalmente, o ambiente ou **estratégia de escalonamento** também é um critério importante, podendo ser dividido em dois métodos: estático ou dinâmico. O escalonamento dinâmico, ao contrário do estático, considera riscos, como avaria de máquinas ou introdução de novas tarefas, o que o torna uma estrutura mais próxima da realidade e mais difícil de resolver (Destouet *et al.*, 2023).

Num processo produtivo definido por uma etapa única, podem ser considerados dois tipos de ambientes: máquina única e máquinas paralelas (Destouet *et al.*, 2023). No escalonamento de máquina única, todas as tarefas são realizadas numa única máquina. O objetivo principal é determinar a sequência mais eficiente para processar todas as tarefas de modo a otimizar uma ou mais medidas de desempenho (Fernanda and Utamima, 2024). O escalonamento de máquinas paralelas consiste na operação simultânea de várias máquinas para realizar a mesma etapa de produção. Estes tipos de problemas são menos investigados pela comunidade científica em relação aos problemas de máquina única, uma vez que as soluções de escalonamento de máquina única constituem uma base fundamental para a resolução de outros desafios de escalonamento (Carrilho, Oliveira and Hamacher, 2024).

Num processo de produção composto por várias etapas, em que as tarefas são divididas em operações que devem ser processadas em máquinas distintas, os ambientes de produção mais estudados são o Flow-Shop e o Job-Shop. Estes dois tipos de ambientes de produção são aplicados em diversos sistemas de fabrico reais (Destouet *et al.*, 2023). Nas seguintes subsecções serão apresentados os problemas de escalonamento Flow-Shop e Job-Shop.

2.1.1 Flow-Shop

Os problemas de escalonamento Flow-Shop têm sido amplamente investigados por vários investigadores, devido às suas diversas aplicações nos setores industrial e económico, sendo amplamente utilizados em indústrias como o fabrico de automóveis, produção de semicondutores e operações de linha de montagem (Komaki, Sheikh and Malakooti, 2018; Mraihi, Driss and EL-Haouzi, 2024).

Um problema de Flow-Shop é composto por um conjunto de tarefas que são executadas por um grupo de máquinas organizadas em série. As tarefas começam na máquina 1, prosseguem para a máquina 2 e percorrem toda a linha de produção passando por todas as máquinas, seguindo a mesma ordem. Uma tarefa só avança para a próxima máquina após a sua conclusão na máquina anterior, sendo que as máquinas só podem processar uma tarefa de cada vez (Ruiz, Pan and Naderi, 2019).

O problema de escalonamento é denominado Permutation Flow-Shop quando a sequência de tarefas é idêntica em todas as máquinas. No entanto, quando a sequência de tarefas difere entre as máquinas, o problema é designado de Non-Permutation Flow-Shop. Os problemas Non-Permutation Flow-Shop apresentam uma maior dificuldade de resolução em comparação com os problemas de Permutation Flow-Shop, devido ao seu maior número de possibilidades que precisam de ser avaliadas para encontrar a solução ótima (Rossit, Tohmé and Frutos, 2018).

O problema Flow-Shop Flexível é uma variação do modelo clássico Flow-Shop. Trata-se de um modelo composto por etapas, em que cada etapa é constituída por um conjunto de máquinas idênticas, dispostas em paralelo. Este tipo de problema também pode ser chamado de Flow-Shop Híbrido (Mraihi, Driss and EL-Haouzi, 2024).

2.1.2 Job-Shop

O problema de escalonamento Job-Shop desempenha um papel importante na investigação operacional e nas ciências de gestão, sendo reconhecido como um dos desafios de otimização combinatória mais clássicos (Xiong *et al.*, 2022). Trata-se de um tema com uma ampla diversidade de áreas de aplicação, como a indústria transformadora e aeroespacial, a logística, os transportes e a medicina (Zhang and Zhu, 2024).

Os problemas de escalonamento Job-Shop têm sido extensivamente estudados na literatura, com destaque para a minimização do *makespan*, que representa o tempo total necessário para completar todas as tarefas, destacando-se como a função objetivo mais analisada neste contexto. A sua importância é amplamente reconhecida, pois impacta diretamente várias medidas-chave do desempenho operacional. De forma distinta, minimizar o *makespan* é equivalente à maximização da utilização média, do número médio de tarefas processadas, da quantidade de tarefas em curso e da taxa de produção (Fuchigami and Rangel, 2018; Fernandez-Viagas, Talens and Prata, 2024).

Um problema de Job-Shop é definido como um ambiente de trabalho que contém um conjunto de máquinas e um conjunto de tarefas, em que cada tarefa é composta por um número de operações que têm de ser executadas numa sequência previamente definida. A cada operação é atribuído uma máquina específica (Xiong *et al.*, 2022). Cada máquina só pode realizar uma operação de cada vez e não pode ser interrompida até à conclusão do processo. Deste modo, a sequência das operações em todas as máquinas deve ser otimizada para atingir os objetivos de otimização definidos (Zhang and Zhu, 2024).

Os problemas de Job-Shop, no contexto real, frequentemente apresentam maior complexidade, exigindo a integração de restrições adicionais, conforme as caraterísticas do ambiente de produção. A inclusão das restrições torna a situação mais desafiadora, como é o caso dos problemas de escalonamento Job-Shop Flexível (Zhang and Zhu, 2024). O Job-Shop Flexível permite que algumas ou todas as operações possam ser executadas por um conjunto de máquinas, em vez de serem restritas a uma única máquina (Destouet *et al.*, 2023).

O problema de escalonamento Job-Shop Dinâmico é uma variação do modelo clássico Job-Shop. Refere-se a um modelo que considera as numerosas alterações dinâmicas que geralmente ocorrem no chão de fábrica. Estas alterações podem ter um impacto significativo nos resultados do escalonamento e, por conseguinte, não devem ser ignoradas. As alterações dinâmicas podem ser classificadas em dois grupos principais: alterações internas e alterações externas. As alterações internas incluem falhas nas máquinas, desvios nos parâmetros dos processos, incerteza nos tempos de máquinas, retrabalho de produtos defeituosos, entre outras situações. No entanto, as alterações externas englobam a colocação de encomendas de emergência, o cancelamento de encomendas, entre outras ocorrências (Zhang and Zhu, 2024).

2.2 Meta-heurísticas

A complexidade dos problemas do mundo real, caraterizados por restrições não lineares, interdependência entre variáveis e uma grande diversidade de soluções possíveis, fundamenta a utilização de técnicas capazes de resolver problemas de otimização complexos em tempos de execução viáveis (Gogna and Tayal, 2013).

As Meta-heurísticas são algoritmos avançados desenvolvidos para resolver uma variedade de problemas de otimização. Tratam-se de técnicas que pertencem aos métodos por aproximação, uma abordagem amplamente utilizada na resolução de problemas de otimização combinatória (Pereira *et al.*, 2021; Hajji, Hamlaoui and Hadda, 2024).

O termo "Meta-heurística" tem origem na combinação de duas palavras gregas, *heuriskein*, que significa descobrir e *meta*, que significa "para além, num nível superior" (Alorf, 2023). As Meta-heurísticas são definidas como processos iterativos de desenvolvimento que direcionam uma ou mais heurísticas subordinadas, recorrendo à combinação inteligente de múltiplos conceitos para explorar e analisar o espaço de pesquisa. Não estando garantida a obtenção da solução ótima, as Meta-heurísticas podem incorporar estratégias de aprendizagem com o intuito de estruturar a informação de forma eficiente, contribuindo para a obtenção de soluções próximas do ótimo (Gogna and Tayal, 2013).

As diversas categorias de Meta-Heurísticas apresentam em comum um conjunto de conceitos fundamentais, nomeadamente (Pereira *et al.*, 2021):

- **Função objetivo:** A função objetivo é um dos elementos mais relevantes na realização das Meta-heurísticas, podendo ser de maximização ou minimização, dependendo do problema a resolver; a função objetivo estabelece a meta a atingir e direciona o processo de pesquisa na procura de soluções ótimas.
- **Solução inicial:** A solução inicial pode ser obtida de forma aleatória ou definida utilizando heurísticas construtivas.
- **Estrutura de vizinhança:** A estrutura de vizinhança é uma etapa também importante, uma vez que define o espaço de pesquisa das soluções.
- **Critério de paragem:** Normalmente, é aplicado um critério de paragem, como o número máximo de iterações ou um número específico de iterações sem melhoria da melhor solução; também é possível implementar um limite de tempo computacional.

Na literatura são apresentadas diversas abordagens relativas à dimensão das Meta-heurísticas, contudo, de forma geral, pode-se afirmar que se classificam em duas categorias: baseada numa única solução e baseada em população (Figura 2). Os algoritmos de Meta-heurísticas baseadas numa única solução utilizam uma solução candidata e melhoram essa solução utilizando a pesquisa local. No entanto, nesta categoria, a solução obtida pode ficar presa em ótimos locais. As Meta-heurísticas baseadas numa única solução incluem, por exemplo, o método Simulated Annealing, Tabu Search e Guided Local Search. As Meta-heurísticas baseadas em população utilizam múltiplas soluções candidatas durante o processo de pesquisa e podem ser divididas em dois grupos: algoritmos Evolutivos (por exemplo, Evolução Diferencial, Algoritmos

Genéticos, entre outras) e algoritmos Swarm Intelligence (por exemplo, Particle Swarm Optimization, Ant Colony Optimization, Artifical Bee Colony, entre outras) (Katoch, Chauhan and Kumar, 2020).

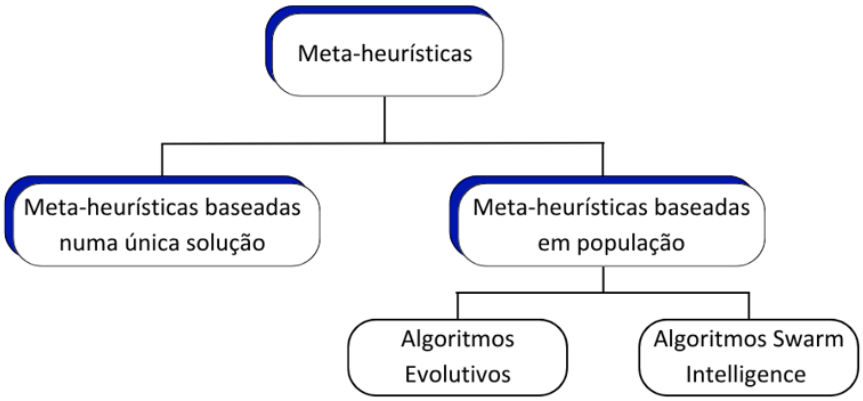


Figura 2 - Classificação das Meta-heurísticas, baseado em Katoch, Chauhan and Kumar (2020)

A seleção da Meta-heurística mais adequada para a resolução de um problema é considerada uma etapa importante. Trata-se de um processo frequentemente considerado difícil que requer uma análise detalhada do tipo de problema e um conhecimento das técnicas disponíveis. Além disso, é difícil eleger uma Meta-heurística como a melhor de todas, uma vez que a sua eficácia depende das propriedades do problema em análise. Cada Meta-heurística possui vantagens e desvantagens (Pereira *et al.*, 2021).

As Meta-heurísticas possuem ainda parâmetros que necessitam de ser determinados, sendo comum o seu ajuste de forma manual por meio de processos de tentativa-erro. A definição de valores adequados para os parâmetros é um processo entediante que tem um efeito significativo no desempenho e é reconhecido como uma tarefa difícil (Pereira, 2014). Como tal, a afinação dos parâmetros é responsável pela eficiência de qualquer algoritmo Meta-heurístico. Conseguir uma afinação ótima dos parâmetros é um dos tópicos mais exigentes, mas ainda pouco estudados, da investigação em Meta-heurísticas (Pereira *et al.*, 2021).

Nas subseções seguintes, é apresentada uma análise de algumas Meta-heurísticas, com destaque particular para as que foram utilizadas no desenvolvimento desta dissertação, nomeadamente Simulated Annealing, Tabu Search, Algoritmos Genéticos, Ant Colony Optimization e Particle Swarm Optimization. Uma descrição minuciosa do problema de afinação de parâmetros das Meta-heurísticas é também efetuada.

2.2.1 Simulated Annealing

O Simulated Annealing foi introduzido pela primeira vez no domínio da otimização por Kirkpatrick, Gelatt and Vecchi (1983). Esta metodologia é fundamentada em conceitos de mecânica estatística e é orientada por uma analogia com o comportamento de sistemas físicos durante o processo de arrefecimento. O algoritmo baseia-se no processo de recozimento, que

envolve o aquecimento de uma substância e o seu arrefecimento gradual para obter uma estrutura cristalina de elevada resistência. O Simulated Annealing é um método de pesquisa de vizinhança sem memória, capaz de evitar a convergência prematura ao evitar os ótimos locais (Gogna and Tayal, 2013; Gallo and Capozzi, 2019). A Tabela 1 apresenta a analogia estabelecida entre os sistemas físicos e o algoritmo de Simulated Annealing.

Tabela 1 - Analogia entre o Sistema Físico e o algoritmo de Simulated Annealing, baseado em Gogna and Tayal (2013)

Sistema Físico	**Simulated Annealing**
Estados	Solução
Energia	Função Objetivo
Estado Fundamental	Solução Ótima
Temperatura	Parâmetros de Controlo
Mudança de Estado	Vizinho

O algoritmo do Simulated Annealing começa com a definição de uma solução inicial S, estabelecida de forma aleatória ou por meio de uma heurística, e com a inicialização do parâmetro de temperatura, T. Em cada iteração é selecionada aleatoriamente uma solução S' que pertence à vizinhança da solução atual S. A solução vizinha S' é avaliada e, se apresentar um desempenho superior à solução S, é aceite como nova solução. Se S' apresentar um desempenho inferior a S, a sua aceitação será decidida com base numa probabilidade, onde $f(S')$ e $f(S)$ representam os valores da função objetivo de S'e S, respetivamente. A medida da probabilidade é definida conforme apresentado na Equação 1 (Boussaïd, Lepagnot and Siarry, 2013).

$$P = exp\left(-\frac{f(S') - f(S)}{T}\right) \quad (1)$$

À medida que o algoritmo progride, a temperatura T é reduzida, o que resulta na diminuição da probabilidade de aceitar movimentos de soluções piores (Boussaïd, Lepagnot and Siarry, 2013).

O Simulated Annealing, com determinadas precauções, pode ser implementado de forma eficaz para alcançar soluções de qualidade (Gogna and Tayal, 2013). Um dos parâmetros básicos nas etapas do desenvolvimento do algoritmo é a seleção da solução inicial. Do ponto de vista teórico, a escolha da solução inicial não influencia a qualidade da solução final, ou seja, a solução tenta convergir para um ótimo global, independentemente da solução inicial selecionada. No entanto, existem exceções, nas quais foi demonstrado que, em alguns casos, o processo de convergência para a solução ótima ocorre de forma mais rápida quando se aplica

como solução inicial uma solução obtida através de uma boa heurística. Contudo, o tempo computacional total para obter uma solução aproximada a partir de uma "boa" solução inicial é frequentemente superior ao tempo necessário para alcançar uma solução aproximada a partir de qualquer solução inicial (Gallo and Capozzi, 2019).

A definição da temperatura inicial constitui um parâmetro de grande relevância na implementação do algoritmo de Simulated Annealing. A temperatura inicial deve ser definida como suficientemente alta. Uma temperatura demasiado baixa pode resultar na limitação do sistema a ótimos locais, enquanto um valor elevado de temperatura pode dificultar a obtenção de uma solução ótima. Além disso, a temperatura deve ser gradualmente reduzida e o sistema deve estabilizar após um número suficiente de iterações a cada temperatura. O modelo de arrefecimento utilizado tem um impacto significativo no desempenho do algoritmo afetando a qualidade da solução final (Gogna and Tayal, 2013).

2.2.2 Tabu Search

O algoritmo Tabu Search, uma técnica de pesquisa local proposta por Glover (1986), utiliza as informações recolhidas durante as iterações, armazenadas numa estrutura de memória, para melhorar a eficácia do processo de pesquisa. A principal caraterística do Tabu Seach consiste na implementação de mecanismos baseados na memória humana (Boussaïd, Lepagnot and Siarry, 2013).

O Tabu Search, assim como o algoritmo Simulated Annealing, aceita soluções que apresentam um desempenho inferior para evitar ótimos locais. Contudo, ao contrário do Simulated Annealing, que realiza uma pesquisa aleatória, o algoritmo Tabu Search explora toda a vizinhança de forma determinística. Uma caraterística única deste algoritmo é a utilização de uma memória de curto prazo, que impede que soluções previamente exploradas sejam novamente aceites, contribuindo para uma obtenção mais rápida da solução ótima (Gogna and Tayal, 2013).

O algoritmo Tabu Search realiza um procedimento iterativo, explorando o espaço de soluções a partir de uma solução inicial e, possui uma lista tabu que contém o registo das soluções anteriormente visitadas. Em cada iteração, novas soluções, denominadas vizinhos, são criadas e avaliadas, sendo que a melhor solução não tabu é selecionada. O melhor vizinho é inserido na lista tabu e, caso esta atinja a sua capacidade máxima, o registo mais antigo é removido. Após um número predeterminado de iterações, o algoritmo retorna a melhor solução encontrada (Gogna and Tayal, 2013; Pereira, 2014; Niroumandrad, Lahrichi and Lodi, 2024).

A memória do processo de pesquisa é definida pelo comprimento da lista tabu. Se a lista for curta, a pesquisa concentrar-se-á em pequenas áreas do espaço. Por outro lado, um comprimento elevado obriga o processo de pesquisa a explorar regiões maiores. O comprimento da lista tabu pode variar durante a pesquisa, o que exige o desenvolvimento de um algoritmo de Tabu Search mais robusto (Boussaïd, Lepagnot and Siarry, 2013).

A eficácia da técnica Tabu Search está diretamente relacionada com a seleção adequada do operador da vizinhança e do espaço de pesquisa. Esta abordagem exige um conhecimento detalhado do problema em análise. Além disso, a lista tabu deve ser cuidadosamente elaborada, com o intuito de maximizar a eficácia da pesquisa, minimizando o tempo de computação e os requisitos de memória (Gogna and Tayal, 2013).

2.2.3 Algoritmos Genéticos

Os Algoritmos Genéticos são uma das primeiras Meta-heurísticas baseadas nos princípios da seleção natural e da evolução. Devido à sua estrutura simples e fácil implementação, destacam-se como uma das estratégias evolucionárias mais bem-sucedidas. O Algoritmo Genético foi desenvolvido no início dos anos 70 na Universidade de Michigan por John Henry Holland e os seus alunos, com o objetivo de estudar o fenómeno da adaptação natural e explorar como este mecanismo poderia ser aplicado a sistemas informáticos para a resolução de problemas complexos (Holland, 1975; Boussaïd, Lepagnot and Siarry, 2013; Fausto *et al.*, 2019).

Os Algoritmos Genéticos consistem em criar uma população inicial, composta por soluções individuais, denominadas cromossomas. A população é submetida a um processo evolutivo que ocorre ao longo de um determinado número de iterações, designadas gerações. Em cada geração, um conjunto de indivíduos (progenitores) é selecionado com base num critério de seleção. Os indivíduos selecionados são cruzados entre si, de acordo com uma taxa de cruzamento previamente definida, originando novos cromossomas, denominados descendentes. Os descendentes são submetidos a uma operação de mutação, mediante uma taxa de mutação, com o intuito de introduzir diversidade no algoritmo. Este processo dá origem a uma nova população que, ao longo de várias gerações, permite ao algoritmo convergir para o melhor cromossoma, o qual se espera que represente a solução ótima do problema (Boussaïd, Lepagnot and Siarry, 2013; Gogna and Tayal, 2013; Desale *et al.*, 2015).

Os Algoritmos Genéticos utilizam diversos operadores durante o processo de pesquisa, destacando-se entre eles a seleção, o cruzamento e a mutação. O operador de seleção desempenha um papel bastante importante no algoritmo, sendo responsável por definir quais os indivíduos que serão incluídos no processo de reprodução. As técnicas de seleção mais conhecidas são a roleta, o torneio e o elitismo. No método da roleta, a probabilidade de seleção de um indivíduo é diretamente proporcional ao seu valor de aptidão. Isto significa que, quanto maior for a aptidão de um indivíduo, maior será a probabilidade de ser escolhido, enquanto indivíduos com menor aptidão terão uma probabilidade reduzida de serem selecionados (Pereira, 2014; Katoch, Chauhan and Kumar, 2020).

O operador de seleção por torneio seleciona aleatoriamente um subconjunto de indivíduos da população, os quais são avaliados com base no seu valor de aptidão, permitindo que os indivíduos com melhor desempenho sejam selecionados para o processo de reprodução (Pereira, 2014; Maier *et al.*, 2019). O operador de seleção por elitismo permite a passagem direta, para a geração seguinte, de um número fixo de indivíduos da população que apresentam

os valores mais elevados de aptidão. Para completar a nova geração, é aplicado outro método de seleção, no qual os indivíduos selecionados são submetidos a operações de cruzamento e mutação (Gogna and Tayal, 2013).

O operador de cruzamento dos Algoritmos Genéticos é responsável pela produção de novos indivíduos (descendentes), a partir da combinação das informações genéticas dos indivíduos previamente selecionados (progenitores). As técnicas de seleção mais conhecidas são o cruzamento de ponto único, o cruzamento de dois pontos e o cruzamento uniforme. O método de cruzamento de ponto único é a abordagem mais simples. Consiste na seleção de um ponto de cruzamento aleatório, no qual a informação genética dos progenitores é trocada a partir desse ponto. O operador de cruzamento de dois pontos implica a seleção aleatória de dois pontos, entre os quais a informação genética dos progenitores é trocada. O método de cruzamento uniforme envolve a troca, entre os progenitores, da informação genética de determinados índices selecionados aleatoriamente (Katoch, Chauhan and Kumar, 2020; Alhijawi and Awajan, 2023). Na Figura 3 são representadas as técnicas de cruzamento mencionadas.

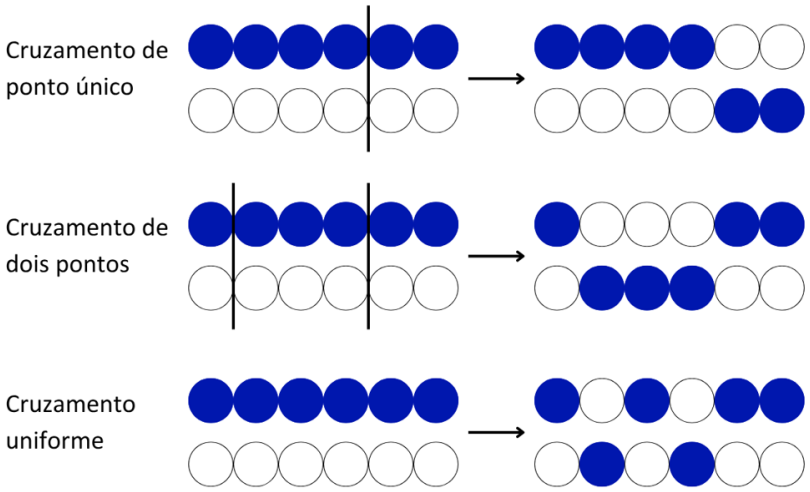


Figura 3 - Técnicas de cruzamento Algoritmos Genéticos, baseado em Alhijawi and Awajan (2023)

O operador de mutação dos Algoritmos Genéticos tem como função introduzir pequenas alterações aleatórias nos indivíduos, com o intuito de promover uma maior diversidade genética na população. A principal finalidade deste operador é evitar que ocorra uma convergência prematura para ótimos locais (Gogna and Tayal, 2013; Pereira, 2014).

A eficácia dos Algoritmos Genéticos depende da seleção da função objetivo (função de aptidão), da dimensão da população e das probabilidades de cruzamento e mutação. Desta forma, a seleção cuidadosa destes parâmetros é fundamental para o desempenho do algoritmo (Gogna and Tayal, 2013).

2.2.4 Ant Colony Optimization

A Ant Colony Optimization é uma Meta-heurística inspirada na natureza, desenvolvida para resolver problemas complexos de otimização combinatória. Esta metodologia foi proposta pela primeira vez por Marco Dorigo em 1992 com o nome de Ant System e fundamenta-se no comportamento natural das formigas. Apesar de serem cegas e pouco inteligentes, as formigas conseguem determinar o caminho mais curto entre uma fonte de alimento e a colónia de origem (Boussaïd, Lepagnot and Siarry, 2013; Fausto *et al.*, 2019; Sadeghian *et al.*, 2023).

No ambiente natural, as formigas deslocam-se de forma aleatória enquanto procuram alimento e, ao identificarem uma fonte adequada, retornam à colónia deixando um rasto de feromona. A feromona é uma substância química que é depositada no caminho quando a formiga se encontra em movimento. As formigas são capazes de se orientar em direção à fonte de alimento previamente localizada, seguindo o percurso marcado pelas feromonas depositadas por elas ou por outras formigas. No entanto, à medida que o tempo passa, a feromona evapora-se. Naturalmente, quanto mais tempo uma formiga demora a percorrer um determinado caminho, mais tempo a feromona tem para se dissipar. Por outro lado, os caminhos mais curtos são percorridos com maior frequência, o que faz com que a densidade da feromona se torne mais elevada em comparação com os percursos mais longos. Neste contexto, caso uma formiga identifique um bom caminho (curto) entre a colónia e a fonte de alimento, é bastante provável que os outros membros sigam o caminho traçado por esta formiga (Radhika and Chaparala, 2018; Fausto *et al.*, 2019).

O algoritmo Ant Colony Optimization começa com a atribuição de um valor inicial à concentração de feromona e a determinação do número de formigas. Durante cada iteração k do algoritmo, cada formiga i elabora o seu caminho (solução) considerando como referência o nível de feromona atual. Após a construção das soluções por todas as formigas da população, o valor da feromona é atualizado, com base nas soluções previamente obtidas. A Equação 2 representa a expressão utilizada para a realização da atualização da feromona (Pereira, 2014; Fausto *et al.*, 2019).

$$t_{(xy)}^{k} = (1 - p) \times t_{(xy)}^{k} + \sum_{i=1}^{N} \Delta t_{i(xy)}^{k} \qquad (2)$$

Na atualização do valor da feromona, xy representa um determinado caminho que conecta os pontos x e y, p é o coeficiente de evaporação da feromona, N indica o número total de formigas na população e $t_{i(xy)}^{k}$ representa a quantidade de feromona depositada pela formiga i no percurso xy. O coeficiente de evaporação da feromona desempenha um papel crucial ao impedir que o algoritmo convirja para ótimos locais (Gogna and Tayal, 2013; Fausto *et al.*, 2019).

Para problemas de seleção, pesquisa e otimização, a Meta-heurística Ant Colony Optimization é utilizada para resolver problemas de otimização contínua, bem como problemas de

otimização de variáveis mistas. O algoritmo Ant Colony Optimization permite a obtenção de resultados num tempo de processamento relativamente aceitável (Kar, 2016).

2.2.5 Particle Swarm Optimization

A Particle Swarm Optimization é uma técnica de pesquisa e otimização computacional, inspirada em fenómenos biológicos e naturais, desenvolvida por Kennedy and Eberhart (1995). Trata-se de uma metodologia que se fundamenta no comportamento coletivo de organismos, como cardumes de peixes, enxames de insetos ou bandos de aves, que trabalham em conjunto para atingir um objetivo (Desale *et al.*, 2015; Kar, 2016).

A Particle Swarm Optimization é um mecanismo baseado em população, mas, ao contrário das outras técnicas, mantém uma única população estática, cujos indivíduos são ajustados com base no espaço de pesquisa. Assim, em vez de se tratar de uma população de indivíduos, as soluções candidatas são denominadas como enxame de partículas. As partículas são criadas de forma estocástica no espaço de pesquisa e cada uma é representada por uma velocidade, uma localização no espaço de pesquisa e a sua melhor posição (Boussaïd, Lepagnot and Siarry, 2013; Gogna and Tayal, 2013; Pereira, 2014).

Na Particle Swarm Optimization, um enxame de partículas movimenta-se num espaço de pesquisa com D dimensões. Cada partícula i é definida por um vetor de posição $X_i = \{x_{i1}, x_{i2}, \dots, x_{iD}\}$, que indica a sua localização na dimensão, e por um vetor de velocidade $V_i = \{v_{i1}, v_{i2}, \dots, v_{iD}\}$. O algoritmo tem início com a atribuição aleatória de uma posição atual e velocidade a todas as partículas. Em cada iteração k, ocorre uma atualização da velocidade e posição da partícula i, sendo influenciada pela melhor posição previamente encontrada pela própria partícula ($pBest$) e pela melhor posição global identificada pelo enxame ($gBest$) (Piotrowski, Napiorkowski and Piotrowska, 2020; Shami *et al.*, 2022). As Equações 3 e 4 correspondem, respetivamente, à atualização da velocidade e da posição da partícula.

$$v_{id}(t+1) = w_k \times v_{id}(t) + c1r1\big(pBest_{id}(t) - x_{id}(t)\big) \quad (3)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (4)$$

Nas expressões de atualização, d pertence ao conjunto $\{1, 2, \dots, D\}$ e w_k representa o peso da inércia, o qual pode depender da iteração ou ser constante durante a procura. Os parâmetros $c1$ e $c2$ são, respetivamente, os coeficientes de aceleração cognitiva e social. O $r1$ e $r2$ são dois valores aleatórios uniformes gerados dentro do intervalo [0; 1]. Este processo repete-se até que o algoritmo atinja o critério de paragem, que pode estar relacionado a um número fixo de iterações (Piotrowski, Napiorkowski and Piotrowska, 2020; Shami *et al.*, 2022). Com o intuito de consolidar o processo da Meta-heurística Particle Swarm Optimization, na Figura 4 é apresentado um fluxograma que ilustra as etapas desta técnica.

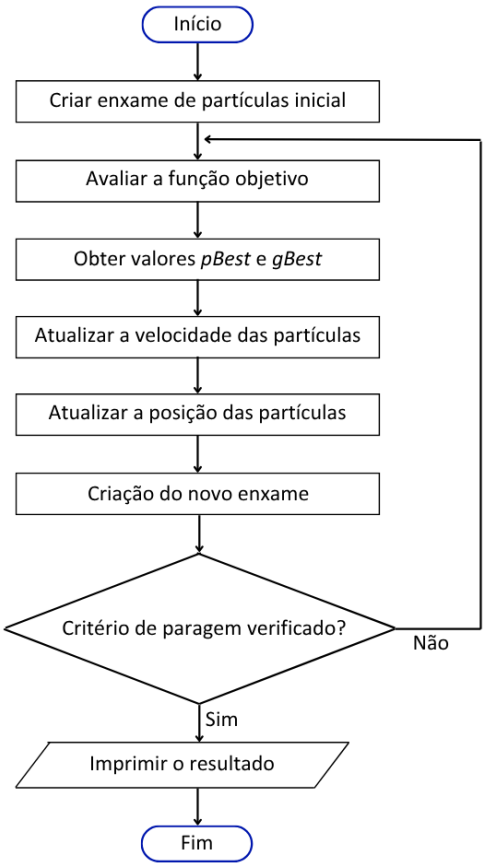


Figura 4 - Fluxograma das etapas de Particle Swarm Optimization, baseado em Radhika and Chaparala (2018)

A Meta-heurística Particle Swarm Optimization é caracterizada por três parâmetros de controlo principais: o peso da inércia, o componente cognitivo e o componente social. O peso da inércia determina a amplitude do espaço de pesquisa. Um valor elevado de inércia incentiva a exploração global, enquanto um valor reduzido promove uma exploração mais local. Os coeficientes de aceleração cognitiva e social orientam a procura do algoritmo para a solução ótima. O desempenho da Particle Swarm Optimization é fortemente influenciado por estes parâmetros, sendo que a obtenção de resultados ótimos depende da configuração adequada dos mesmos (Boussaïd, Lepagnot and Siarry, 2013; Gogna and Tayal, 2013; Shami *et al.*, 2022).

A Particle Swarm Optimization destaca-se como uma abordagem atrativa devido à sua estrutura simples e fácil implementação e, como referido anteriormente, possui apenas três parâmetros de controlo. Adicionalmente, é uma técnica que possui flexibilidade, visto que permite a sua combinação com outros algoritmos de otimização (Shami *et al.*, 2022).

2.2.6 Artificial Bee Colony

A Artificial Bee Colony é um algoritmo de otimização estocástica de estrutura simples e baseado em população. Esta Meta-heurística foi proposta por Karaboga (2005) e fundamenta-se na cooperação social das abelhas durante a procura de recursos alimentares. No algoritmo, as

soluções candidatas são representadas pelas abelhas, que exploram as fontes alimentares, e cada solução corresponde a um recurso alimentar específico, enquanto a qualidade de cada solução é determinada pela quantidade de alimento presente no recurso (Desale *et al.*, 2015; Dokeroglu, Deniz and Kiziloz, 2022).

Uma colónia de abelhas é composta por uma fêmea reprodutora, a rainha, alguns machos, conhecidos como zangões, e várias fêmeas estéreis, chamadas trabalhadoras. A rainha é a única fêmea responsável pelo acasalamento com os zangões e pela produção de um número significativo de abelhas, iniciando assim uma nova geração. Os ovos fertilizados dão origem a fêmeas, abelhas trabalhadoras, e os ovos não fertilizados tornam-se zangões (Boussaïd, Lepagnot and Siarry, 2013).

As abelhas trabalhadoras, em certas ocasiões, podem estar empregadas ou desempregadas. As abelhas empregadas dedicam-se à exploração de uma fonte de alimento, transportando consigo informações detalhadas sobre a fonte que são posteriormente partilhadas com a colónia. As abelhas desempregadas dividem-se em dois tipos: exploradoras e espectadoras. As abelhas exploradoras são responsáveis por identificar novas fontes de alimento, enquanto as abelhas espectadoras permanecem na colónia, aguardando a partilha de informações pelas abelhas empregadas para apoiar na exploração dos recursos alimentares (Pereira, 2014; Desale *et al.*, 2015; Dokeroglu, Deniz and Kiziloz, 2022).

As abelhas são insetos que se destacam pelo seu comportamento coletivo, particularmente na obtenção de alimento, que é a atividade mais importante da colónia. O processo começa com a exploração de fontes de comida pelas abelhas exploradoras, com o intuito de encontrar pólen, néctar ou própolis. A abelha exploradora, após a identificação de um recurso alimentar e descarregar o alimento na colmeia, torna-se uma abelha empregada. A partilha de informação sobre a fonte de alimento é realizada através da execução de uma serie de movimentos, frequentemente denominada como uma dança (*waggle dance*). As abelhas realizam os movimentos tentando persuadir as abelhas espectadoras a explorar a fonte. Cada colmeia possui uma área destinada à execução desta dança, onde são transmitidos os detalhes sobre a distância e a direção em relação à colmeia, bem como a qualidade da fonte de alimento. As abelhas espectadoras analisam várias danças e decidem qual fonte de alimento que pretendem explorar. Após selecionarem e localizarem o recurso de alimento, as abelhas espectadoras tornam-se abelhas empregadas. Ao esgotar-se a fonte de alimento, as abelhas empregadas tornam-se abelhas exploradoras, dedicando-se à exploração de novos recursos alimentares (Boussaïd, Lepagnot and Siarry, 2013; Pereira, 2014; Fausto *et al.*, 2019).

O algoritmo Artifical Bee Colony, ao simular o comportamento inteligente das colónias de abelhas reais, tenta encontrar uma solução global ótima ou quase ótima para os problemas de otimização. Nesta metodologia, o número de fontes de alimento é igual ao número de abelhas empregadas (Kiran and Findik, 2015; Li and Yang, 2016).

A Artifical Bee Colony é composta por uma população de tamanho SN. No contexto do problema de otimização, cada fonte de alimento é representada por num vetor de parâmetros

de dimensão D, que codifica a solução candidata, sendo definido como $X_i = \{x_{1i}, x_{2i}, \dots, x_{Di}\}$, $i = 1, 2, \dots, SN$. No algoritmo, a população inicial de fontes de alimento pode ser criada de forma aleatória. Para assegurar uma ampla cobertura do espaço de pesquisa, as fontes de alimento iniciais são distribuídas uniformemente, respeitando os limites mínimo e máximo predefinidos pelos parâmetros, ou seja, $X_{min} = \{x_{1min}, x_{2min}, \dots, x_{Dmin}\}$ e $X_{max} = \{x_{1max}, x_{2max}, \dots, x_{Dmax}\}$ (Li and Yang, 2016). Para o parâmetro j na fonte de alimentação i, o valor inicial x_i^j é:

$$x_i^j = x_{min}^j + rand(0,1) \times \left(x_{max}^j - x_{min}^j\right), \qquad j = 1, 2, \dots, D \tag{5}$$

Após explorar o espaço, as abelhas empregadas regressam à colmeia e transmitem informações sobre a fonte alimentar às abelhas espectadoras. As abelhas espectadoras são alocadas nas fontes de alimento com base em uma probabilidade proporcional à quantidade de alimento, ou seja, à qualidade da solução. A probabilidade é calculada conforme apresentado na Equação 6, na qual fit_i indica o valor de aptidão da fonte de alimento i (Li and Yang, 2016).

$$p_i = \frac{fit_i}{\sum_{j=1}^{SN} fit_j} \tag{6}$$

Uma vez escolhida a fonte de alimento, as abelhas espectadoras procedem à seleção de um vizinho para explorar. Uma das maneiras de determinar um vizinho pode ser realizada por meio da Equação 7. Nesta equação, $k \in \{1, 2, \dots, SN\}$ representa uma fonte de alimento vizinha selecionada aleatoriamente, onde k deve ser diferente de i, e $\emptyset_i^j$ é um valor aleatório no intervalo [-1; 1]. Após a obtenção do vizinho, este é avaliado e comparado com a solução anterior. Se a qualidade do vizinho for superior, este substituirá a solução na população (Pereira, 2014; Li and Yang, 2016).

$$v_i^j = x_i^j + \emptyset_i^j \times \left(x_i^j - x_k^j\right) \tag{7}$$

No algoritmo Artifical Bee Colony é possível que uma fonte de alimento não possa ser melhorada. Neste caso, considera-se que a fonte de alimento está abandonada e deve ser removida da população. Para assegurar a diversidade populacional, uma abelha exploradora é designada para gerar uma fonte de alimento, que substituirá a fonte abandonada. Para executar este procedimento, é atribuído a cada fonte de alimento i um parâmetro denominado $limite_i$, que contabiliza o número de tentativas falhadas das abelhas de melhorar a qualidade. Quando o $limite_i$ atinge um determinado valor, isso indica que a fonte de alimento i se

esgotou, sendo então classificada como uma fonte abandonada e substituída por uma nova fonte de alimento (Li and Yang, 2016).

O desempenho da pesquisa local da Meta-heurística Artificial Bee Colony pode ser atribuído aos mecanismos de exploração da vizinhança e à seleção "gananciosa" efetuada à volta da fonte de alimento, enquanto o desempenho da pesquisa global está predominantemente associado aos atributos de diversificação das abelhas exploradoras (Fausto *et al.*, 2019).

2.2.7 O problema de afinação de parâmetros

As Meta-heurísticas são metodologias de alto nível que, geralmente, não são desenvolvidas para se adaptar a problemas específicos, permitindo a sua implementação em diversas situações (Huang, Li and Yao, 2020). Trata-se de uma técnica bastante útil para obter soluções que requerem pouco tempo de computação e, por vezes, pode mesmo chegar a soluções ótimas (Pereira *et al.*, 2021).

As Meta-heurísticas, independentemente de serem técnicas de aproximação para problemas de otimização complexos, contêm dificuldades na sua implementação. Um dos principais obstáculos está relacionado ao impacto da parametrização no desempenho do algoritmo, uma vez que o processo de parametrização compromete o desempenho e a eficácia da abordagem, pois os parâmetros controlam o comportamento dos mecanismos das Meta-heurísticas. Para alcançar um desempenho elevado, os parâmetros do algoritmo devem estar corretamente definidos. A parametrização é o processo que permite que uma Meta-heurística seja adaptada a um problema, ou mesmo a uma instância do problema. Embora a literatura apresente diversas sugestões sobre os valores dos parâmetros para um conjunto semelhante de instâncias de problemas, estas configurações nem sempre são as mais adequadas quando se resolvem instâncias de um problema em questão. Na realidade, a definição de parâmetros não é uma tarefa simples, sendo necessário que os investigadores ajustem os parâmetros sempre que lidam com novas instâncias (Huang, Li and Yao, 2020; Karimi-Mamaghan *et al.*, 2022; Santos, Madureira and Varela, 2022).

A parametrização, embora tenha impacto no desempenho, ainda é frequentemente realizada de forma empírica, com base no conhecimento prévio do utilizador sobre uma determinada Meta-heurística. Apesar de existirem diversos métodos de parametrização, a maioria exige um esforço considerável ou envolve a aplicação de técnicas complexas. A resolução de problemas complexos exige a aplicação de uma Meta-heurística e, adicionalmente, a dedicação considerável de tempo para a realização do processo de parametrização (Santos, Madureira and Varela, 2022).

A parametrização dos algoritmos é classificada em duas categorias: ***offline*** e ***online*** (Figura 5). Na parametrização *offline*, os valores dos parâmetros são identificados antes da aplicação do algoritmo. Neste caso, os parâmetros permanecem inalterados durante a execução da Meta-heurística. Na abordagem *online*, os parâmetros são controlados e atualizados de forma

dinâmica ou adaptativa durante a execução do algoritmo (Huang, Li and Yao, 2020; Pereira *et al.*, 2021).

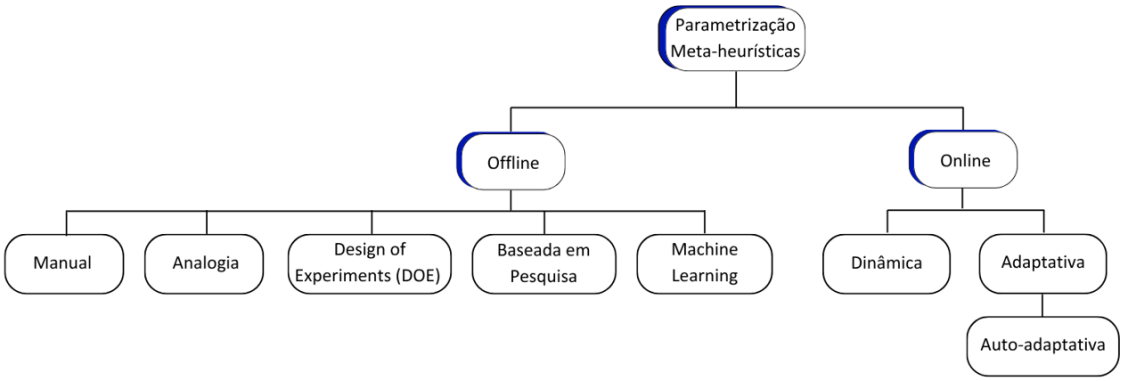


Figura 5 - Parametrização de Meta-heurísticas, baseado em Pereira *et al.* (2021); Santos, Madureira and Varela (2022)

Os processos de parametrização *offline* têm como objetivo avaliar o seu desempenho numa instância ou conjunto de instâncias, utilizando diferentes configurações de parâmetros. Para tal, é necessário realizar um elevado número de execuções do algoritmo o que torna o processo moroso, representando a principal desvantagem da parametrização *offline*. Contudo, o caráter abrangente desta abordagem permite que um método eficiente de afinação seja aplicado à configuração de parâmetros de diversas Meta-heurísticas diferentes. No entanto, a falta do caráter abrangente é a principal desvantagem da parametrização *online*. Além disso, para desenvolver adequadamente estratégias de controlo de parâmetros *online*, é necessário ter uma noção clara de como ajustar o(s) parâmetros(s) durante a execução para obter um bom desempenho. Este processo exige uma compreensão dos valores adequados dos parâmetros nas diferentes fases de execução do algoritmo e envolve a utilização de informações históricas geradas no decorrer do processo iterativo (Huang, Li and Yao, 2020).

A parametrização *offline* pode ser dividida em métodos de afinação manual, por analogia, por Design of Experiments (DOE) e baseada em pesquisa (Montero, Riff and Neveu, 2014; Santos, Madureira and Varela, 2022). Contudo, além destas quatro abordagens, diversos autores destacam o uso de Machine Learning como uma técnica eficaz na parametrização *offline* (Pereira, 2014; Huang, Li and Yao, 2020; Pereira *et al.*, 2021; Karimi-Mamaghan *et al.*, 2022).

O método de afinação manual envolve a modificação iterativa dos parâmetros, exigindo que o utilizador possua um conhecimento aprofundado da Meta-heurística utilizada. Trata-se de um procedimento muito simples, no qual o utilizador executa a Meta-heurística com parâmetros iniciais, os quais são ajustados um a um para melhorar o desempenho, até que o utilizador esteja satisfeito com o resultado obtido. A parametrização manual é um procedimento trabalhoso e pode demorar algum tempo a ser concluído. É considerada a abordagem típica de afinação de parâmetros nos trabalhos de investigação mais antigos na área das Meta-heurísticas (Montero, Riff and Neveu, 2014; Santos, Madureira and Varela, 2022).

Na parametrização por analogia, os utilizadores procuram uma implementação eficaz da Meta-heurística e replicam os parâmetros. Nesta situação, antes de executar o algoritmo, o utilizador irá procurar implementações bem-sucedidas da Meta-heurística, identificará os parâmetros que foram utilizados e aplicará esses parâmetros no algoritmo (Santos, Madureira and Varela, 2022).

O método de afinação por Design of Experiments (DOE) envolve a escolha dos parâmetros pelos utilizadores por meio de múltiplos ensaios experimentais. Neste contexto, antes de executar a Meta-heurística, o utilizador realiza vários testes de parametrização, de seguida, escolhe os parâmetros com melhor desempenho e aplica-os ao algoritmo. Antes da aplicação da parametrização por DOE, é fundamental considerar os fatores que representam as variações dos parâmetros, bem como os níveis que correspondem aos diferentes valores de parâmetros (que podem ser quantitativos ou qualitativos). A principal desvantagem desta abordagem é o elevado custo computacional, particularmente quando existe um grande número de parâmetros e os respetivos domínios de valores são amplos, exigindo um grande número de experiências (Pereira *et al.*, 2021; Santos, Madureira and Varela, 2022).

A parametrização baseada em pesquisa é abordada como um problema de meta-otimização. A meta-otimização é composta por dois níveis: a meta-nível e o nível base. Na meta-nível, as soluções representam os parâmetros a otimizar, como a taxa de arrefecimento no Simulated Annealing e as taxas de cruzamento e mutação nos Algoritmos Genéticos. Neste nível, a função objetivo de uma solução corresponde à melhor solução encontrada para as Meta-heurísticas. Assim, cada solução na meta-nível corresponde a uma Meta-heurística independente no nível base. Em termos mais simples, na parametrização baseada em pesquisa, o espaço dos parâmetros é explorado por uma técnica aproximada, como a pesquisa local ou uma Meta-heurística (Pereira *et al.*, 2021; Santos, Madureira and Varela, 2022).

A parametrização por Machine Learning pode utilizar técnicas como Regressão Logística e Support Vector Machines para prever o desempenho de um determinado conjunto de parâmetros, com base em um conjunto de instâncias de treino. Este processo pode ser dividido em duas etapas: a fase de afinação e a fase de teste. Na fase de afinação, deve ser estabelecida uma configuração dos parâmetros que otimize a medida de desempenho do algoritmo, com base nas instâncias de treino, as quais são representativas do problema a ser abordado. Posteriormente, na fase de teste, a configuração obtida é implementada para resolver instâncias nunca vistas. O objetivo da configuração do algoritmo é encontrar uma definição apropriada dos parâmetros na fase de afinação, de modo a maximizar o desempenho do algoritmo nas instâncias que serão apresentadas durante a fase de teste (Huang, Li and Yao, 2020; Karimi-Mamaghan *et al.*, 2022).

Em relação à parametrização *online*, esta pode ser classificada em dinâmica ou adaptativa. As técnicas dinâmicas não consideram o processo de pesquisa, em vez disso, alteram os valores dos parâmetros de forma aleatória ou determinística. As técnicas adaptativas utilizam memória ou regras pré-estabelecidas para ajustar os valores dos parâmetros de acordo com o processo de pesquisa. As técnicas auto-adapativas, uma subclasse dos métodos adaptativos, são

frequentemente utilizadas na comunidade de computação. Trata-se de uma abordagem que consiste na evolução dos parâmetros durante o processo de pesquisa (Pereira *et al.*, 2021).

2.3 Machine Learning

A Machine Learning é um subdomínio da Inteligência Artificial que consegue transformar informações em conhecimento. Embora a Inteligência Artificial e a Machine Learning sejam frequentemente utilizadas em conjunto, são dois conceitos diferentes. A Inteligência Artificial é um conceito amplo – máquinas que tomam decisões, aprendem novas habilidades e resolvem problemas da mesma forma que os humanos – e a Machine Learning é um subconjunto de Inteligência Artificial que permite que sistemas inteligentes aprendam coisas novas de forma independente a partir dos dados (Hiran et al., 2021).

O objetivo da Machine Learning é identificar padrões úteis e relevantes em conjuntos de dados de treino compostos por várias amostras. Carateriza-se por fornecer aos sistemas a capacidade de aprender e melhorar automaticamente a partir da experiência, sem ser especificamente programado. Esta metodologia cresceu rapidamente nos últimos anos no contexto de análise de dados e computação, permitindo que as aplicações funcionem de maneira inteligente (Sarker, 2021; Talbi, 2021).

A qualidade de uma solução de Machine Learning depende da natureza e caraterísticas dos dados e do desempenho dos algoritmos de aprendizagem. Esta metodologia tem sido utilizada em diversos setores, como robótica, processamento de linguagem natural, recomendação de produtos e diagnósticos de médicos (Ray, 2019; Sarker, 2021).

A Machine Learning dedica-se à análise e ao desenvolvimento de algoritmos que aprendem a partir de dados brutos, treinam o sistema e fazem previsões com bases nesses dados de treino. A ideia principal é a elaboração de algoritmos que, por exemplo, com a utilização da análise matemática, possam receber dados de entrada (*input*) e prever a saída (*output*), enquanto avaliam os resultados à medida que novos dados se tornam disponíveis. De forma a tornar o seu funcionamento mais claro, a Machine Learning pode ser dividida em seis etapas (Chauhan and Singh, 2019; Hiran *et al.*, 2021):

- **Recolha de dados:** A primeira etapa é crucial, uma vez que a qualidade e quantidade dos dados recolhidos terão um impacto direto no desempenho do modelo. A fase inicial na criação do modelo consistirá na recolha de dados relevantes que possam ser utilizados para treinar o modelo.
- **Preparação dos dados:** Nesta fase, os dados são devidamente preparados para serem utilizados para treinar o modelo. Este processo envolve a limpeza dos dados, o que inclui, por exemplo, a remoção dos dados duplicados, o tratamento de valores ausentes e a correção de erros. Além disso, nesta etapa, é necessário realizar a divisão do conjunto de dados em dois grupos: treino e teste. O conjunto de treino, geralmente representando 80% dos dados, será utilizado para treinar o modelo, e o conjunto de teste, tipicamente composto por 20% dos dados, será utilizado para avaliar o desempenho do modelo.

- **Escolha do Modelo:** Esta etapa consiste na seleção do modelo que se revela mais adequado ao trabalho que está a ser desenvolvido. Existem diversos modelos que podem ser aplicados, dependendo das necessidades específicas de cada aplicação.
- **Treinar o Modelo:** Nesta etapa, a maior parte do processo de aprendizagem é realizado. O processo de treino consiste em definir valores aleatórios, digamos X e Y, para o modelo prever o resultado utilizando estes valores, compará-lo com a previsão do modelo e, em seguida, ajustar os valores para que correspondem às previsões feitas anteriormente.
- **Avaliação:** Após a fase de treino do modelo, é necessário testar o modelo para ver se funciona bem em cenários práticos. Desta forma, o conjunto de dados de teste é utilizado para verificar o desempenho do sistema.
- **Previsões:** A etapa final do processo é a previsão, fase em que o modelo é considerado preparado para aplicações práticas.

O desenvolvimento de modelos eficazes em diferentes áreas de aplicação requer a utilização de técnicas apropriadas de Machine Learning, cuja seleção deve ser fundamentada na natureza dos dados e nos resultados esperados. Assim, os algoritmos de Machine Learning dividem-se principalmente em quatros categorias (Sarker, 2021): aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço (Figura 6). Nas próximas subsecções é apresentado uma análise detalhada de cada categoria de Machine Learning.

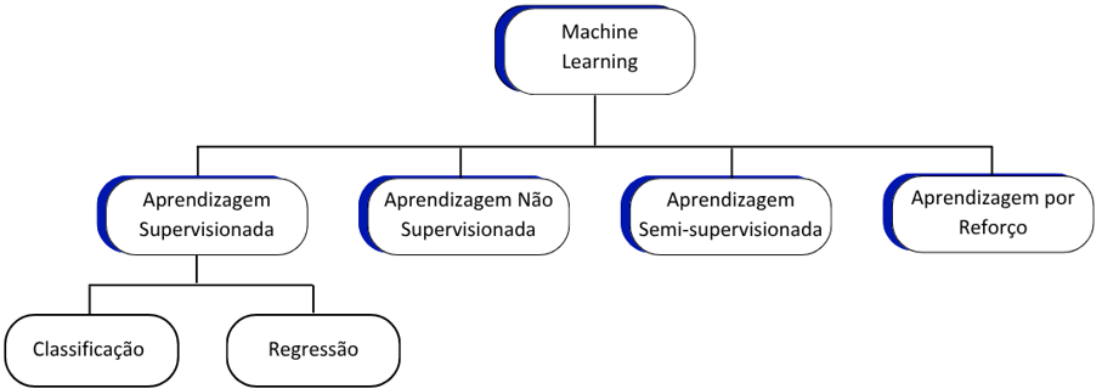


Figura 6 - Classificação de Machine Learning, baseado em Sarker (2021)

2.3.1 Aprendizagem supervisionada

A aprendizagem supervisionada, reconhecida como a principal metodologia de Machine Learning, consiste em algoritmos que se desenvolvem à medida que adquirem experiência (Figura 7). As variáveis de entrada (*input*) e as varáveis de saída (*output*) são conhecidas e encontram-se disponíveis no conjunto de dados da instância (Hiran *et al.*, 2021; Telikani *et al.*, 2021).

A aprendizagem supervisionada tenta encontrar relações entre o conjunto de dados de entrada e os dados de saída, durante o processo de treino. Este procedimento permite a previsão dos

dados de saída quando novos dados de entrada são aplicados (Telikani *et al.*, 2021; Karimi-Mamaghan *et al.*, 2022).

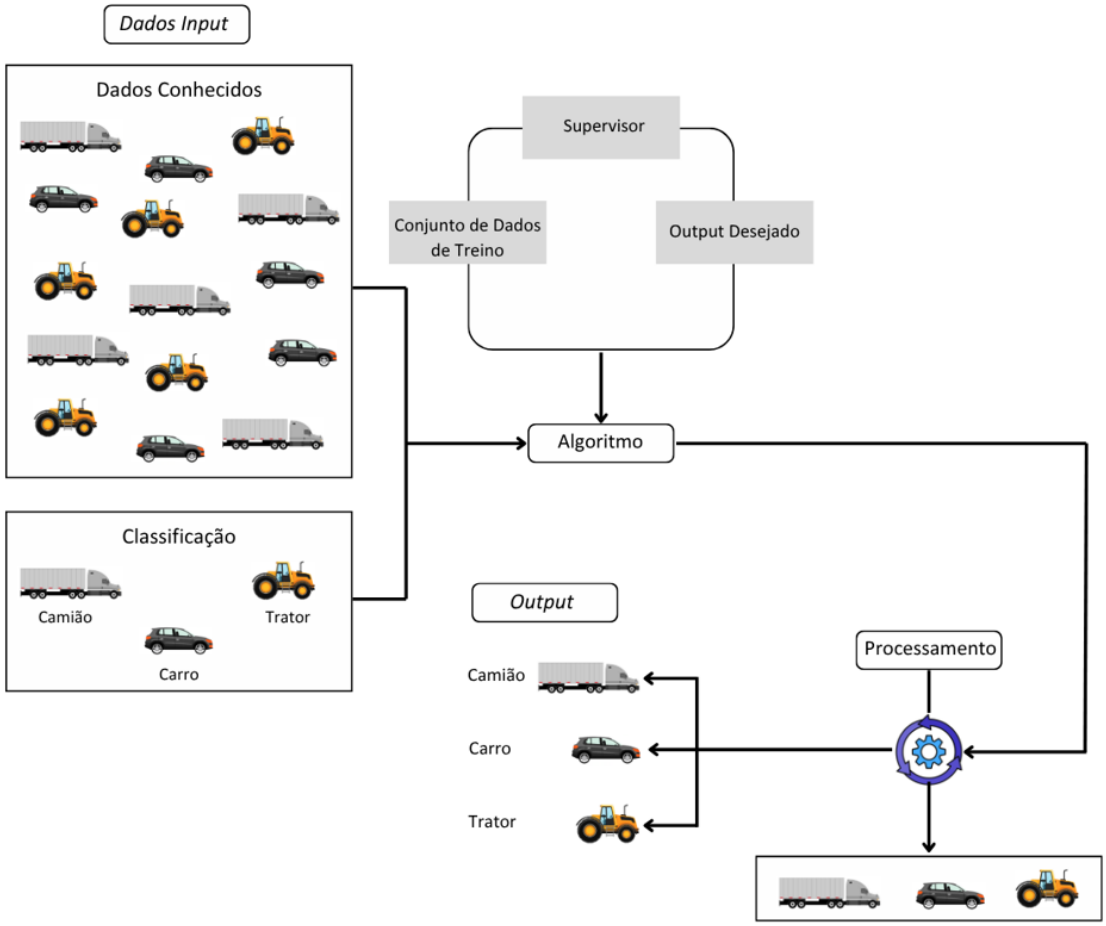


Figura 7 - Aprendizagem Supervisionada

Os algoritmos de aprendizagem supervisionada podem ser divididos em duas categorias: classificação e regressão (Figura 8). A classificação é o processo de agrupar os dados de saída em diferentes classes com base numa ou mais variáveis de entrada. Esta técnica é utilizada quando o valor da variável de saída é discreto ou categórico. A regressão é utilizada quando existe uma relação entre a variável de entrada e a variável de saída. Esta abordagem é aplicada quando o valor da variável de saída é contínuo ou real. O objetivo da regressão consiste em prever um valor o mais próximo possível do valor de saída real, sendo a avaliação efetuada através do cálculo do erro. Quanto menor for o erro, maior será a precisão do modelo de regressão (Hiran *et al.*, 2021).

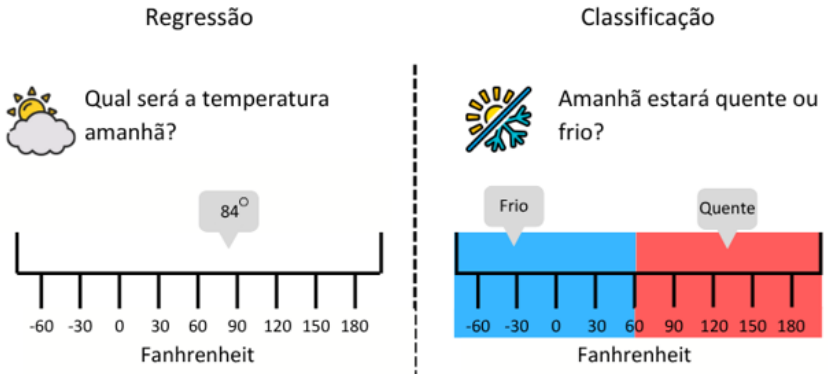


Figura 8 - Aprendizagem Supervisionada (regressão e classificação)

2.3.2 Aprendizagem não supervisionada

Os algoritmos de aprendizagem não supervisionada são utilizados quando os dados não são classificados (Figura 9). Neste caso, os valores das variáveis de entrada são conhecidos, mas não existem valores associados às variáveis de saída (Karimi-Mamaghan *et al.*, 2022).

A aprendizagem não supervisionada consiste em identificar, no conjunto de dados, informações estruturais relevantes, como grupos de elementos que partilham propriedades comuns ou representações de dados que são adaptadas de um espaço de elevada dimensão para um espaço de menor dimensão. Neste contexto, esta abordagem é aplicada em tarefas como o agrupamento de dados com base em semelhanças, sendo igualmente implementada em problemas de redução de dimensionalidade, nos quais as principais características dos dados são extraídas (Verbraeken *et al.*, 2020; Janiesch, Zschech and Heinrich, 2021).

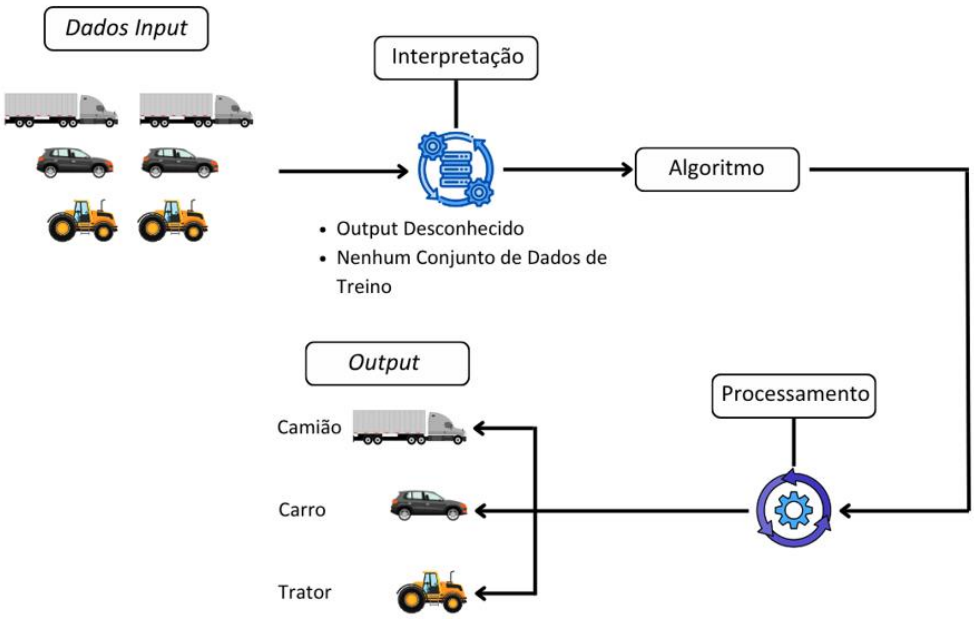


Figura 9 - Aprendizagem Não Supervisionada

2.3.3 Aprendizagem semi-supervisionada

A aprendizagem semi-supervisionada pode ser descrita como uma combinação dos métodos supervisionados e não supervisionados previamente mencionados, uma vez que opera tanto com dados conhecidos como não conhecidos (Figura 10). No contexto prático, os dados conhecidos podem ser escassos em vários contextos e os dados não conhecidos são numerosos, pelo que a aprendizagem semi-supervisionada é uma abordagem útil (Sarker, 2021).

Os algoritmos de aprendizagem semi-supervisionada, normalmente, tentam melhorar o desempenho de uma das tarefas (supervisionada ou não supervisionada) ao utilizar informações tipicamente associadas à outra (van Engelen and Hoos, 2019). Esta metodologia é aplicada em diversas áreas, tais como a tradução automática, deteção de fraudes e a classificação de textos (Sarker, 2021).

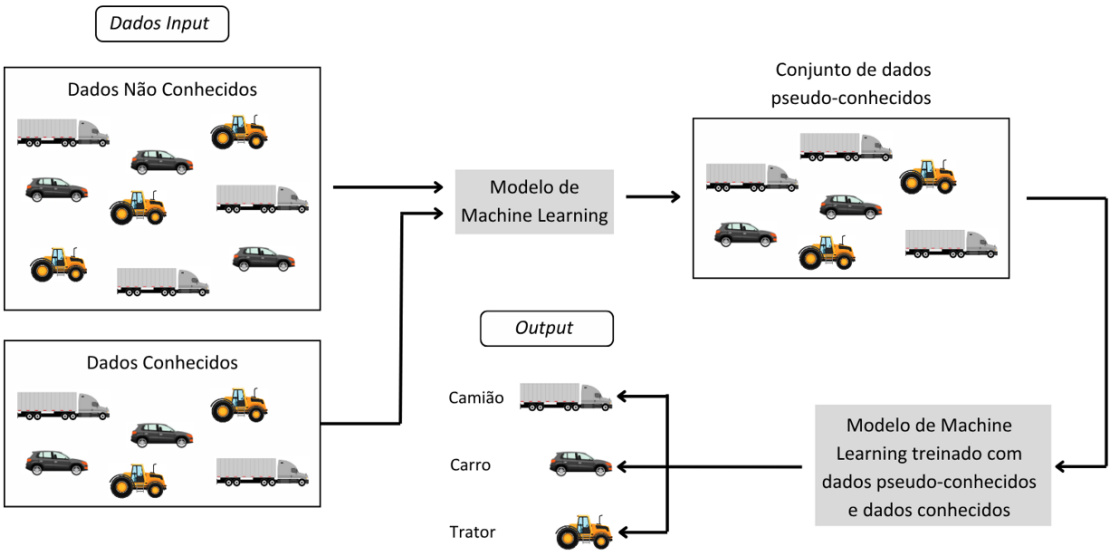


Figura 10 - Aprendizagem Semi-supervisionada

2.3.4 Aprendizagem por reforço

A aprendizagem por reforço é uma metodologia que possibilita a um agente aprender por meio de tentativa-erro, num ambiente iterativo, utilizando informações obtidas a partir das suas ações e experiências (Figura 11). A aprendizagem por reforço baseia-se num agente autónomo, que pode ser uma pessoa, um animal ou um robô, que interage com um ambiente incerto, com o objetivo de maximizar uma recompensa numérica. Porém, a recompensa não é dada imediatamente após a ação, mas sim após uma sucessão de ações que provocam mudanças graduais no ambiente do agente. Para maximizar a pontuação final, o agente tem de aderir a uma política, formada por regras, estratégias e critérios (Sarker, 2021; Telikani *et al.*, 2021).

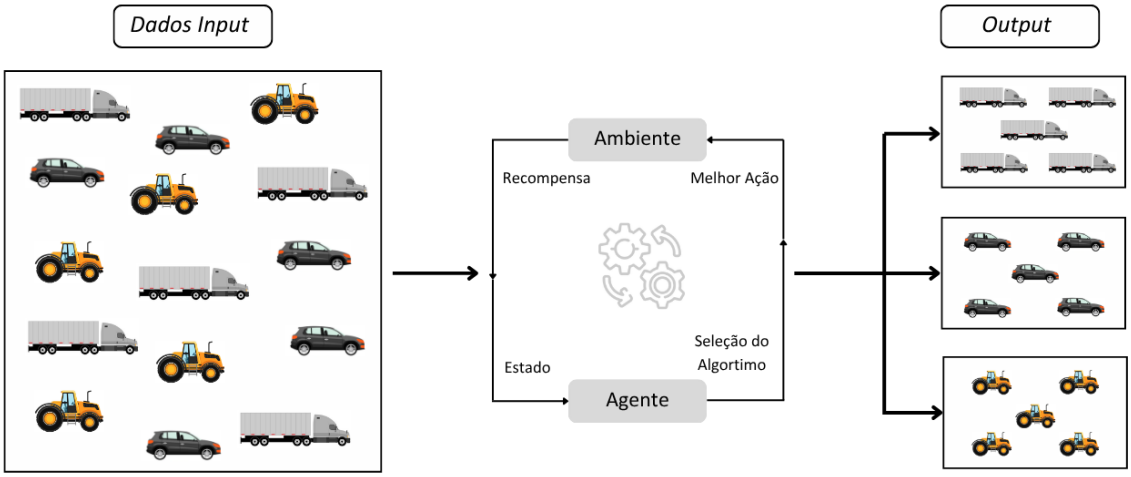


Figura 11 - Aprendizagem por Reforço

2.4 Algoritmos de Machine Learning

Esta secção tem como intuito fornecer uma descrição de alguns algoritmos de aprendizagem supervisionada de Machine Learning. Os algoritmos apresentados são as metodologias que foram consideradas no desenvolvimento desta dissertação.

2.4.1 Árvores de Decisão

A Árvore de Decisão é uma abordagem de aprendizagem supervisionada para resolver problemas de classificação e regressão, dividindo continuamente os dados com base num determinado parâmetro. Trata-se de um método eficaz, utilizado em vários domínios, para resolver problemas científicos e de engenharia (Ray, 2019; Singh Kushwah *et al.*, 2022). Na Figura 12 é apresentado um exemplo de Árvore de Decisão que mostra o processo de previsão meteorológica.

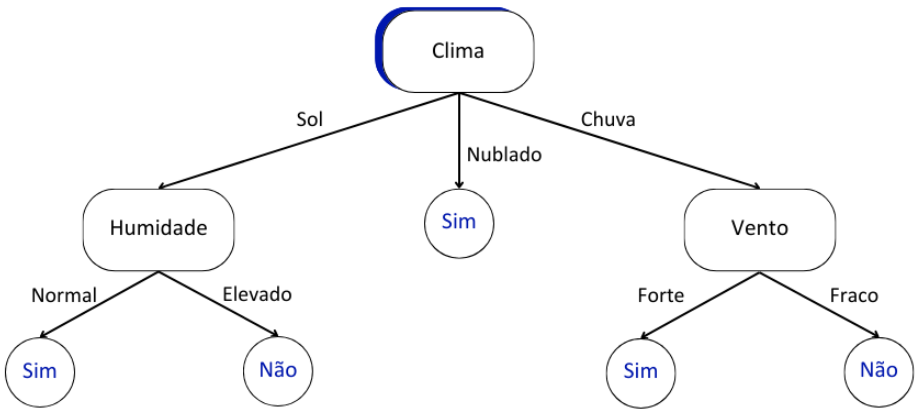


Figura 12 - Exemplo Árvore de Decisão, baseado em Jadhav and Channe (2016)

A Árvore de Decisão é composta por nós internos (condição de teste numa variável), ramos (resultado da condição de teste e responsável pela ligação entre nós) e nós folha (previsão do resultado). A construção da Árvore de Decisão é efetuada numa abordagem de cima para baixo (*top-down*), iniciando o processo no nó raiz, primeiro nó onde ocorre a primeira divisão. A árvore é recursivamente particionada até atingir um nó terminal (também designado de nó folha) (Jadhav and Channe, 2016; Singh Kushwah *et al.*, 2022).

O algoritmo Árvore de Decisão tem como principal objetivo construir um modelo de treino que possa ser utilizado para prever a classe ou valor de variáveis-alvo através da aprendizagem de regras de decisão, inferidas a partir dos dados de treino. Carateriza-se por ser uma abordagem de fácil interpretação e tem a capacidade de tratar valores categóricos e quantitativos. Contudo, o tamanho da Árvore de Decisão, em algumas situações, pode ser difícil de controlar e a complexidade computacional da técnica aumenta quando existe um maior número de amostras de treino (Ray, 2019; Sivananda and Kumar, 2024).

2.4.2 Support Vector Machines

As Support Vector Machines são modelos de aprendizagem supervisionada com algoritmos de aprendizagem associados que analisam dados utilizados para classificação e análise de regressão. A principal motivação das Support Vector Machines é separar várias classes do conjunto de treino com uma superfície que maximiza as margens entre elas (Cervantes *et al.*, 2020; Mahesh, 2020).

Na metodologia Support Vector Machines é fundamental definir um hiperplano que constitui o limite de decisão. Quando existe um conjunto de variáveis que pertencem a classes diferentes é necessário um plano de decisão para os separar. O objetivo principal é obter a maior margem possível, criando a maior distância entre cada classe e o hiperplano de separação. Quanto maior for a margem, menor será o erro de generalização do modelo (Ray, 2019; Thomas Rincy and Gupta, 2020). As Support Vector Machines podem ser classificadas em lineares e não lineares (Figura 13).

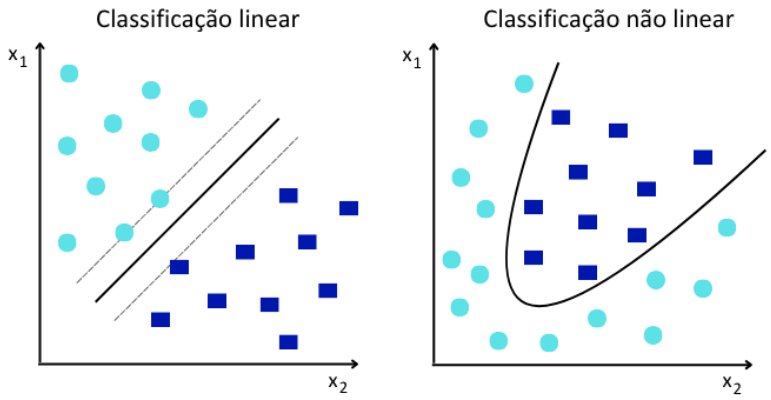


Figura 13 - Support Vector Machines (classificação linear e classificação não linear), baseado em Cervantes *et al.* (2020); Mahesh (2020)

Na classificação não linear torna-se necessário a utilização de técnicas denominadas de truques de *kernel*, que consistem essencialmente em funções matemáticas complexas, capazes de auxiliar na separação das variáveis em classes diferentes (Ray, 2019; Mahesh, 2020).

As Support Vector Machines têm a capacidade de evitar ótimos locais e podem ser aplicadas a dados de elevada dimensão. No entanto, o seu desempenho diminui perante um grande conjunto de dados devido ao aumento do tempo de formação. A técnica Support Vector Machines tem aplicação prática na deteção de fraudes em cartões de crédito, no reconhecimento da escrita manual, na classificação de texto, entre outros (Ray, 2019).

2.4.3 Naive Bayes

A Naive Bayes é uma técnica de classificação baseada no Teorema de Bayes e carateriza-se por ser uma metodologia que assume que as variáveis são condicionalmente independentes, dada a classe. Em termos simples, a Naive Bayes parte do princípio de que a presença de uma determinada variável numa classe não está relacionada com a presença de qualquer outra variável. No entanto, no contexto do mundo real, o pressuposto que todas as variáveis são independentes uma das outras dificilmente é verdadeiro. Apesar disso, em muitas aplicações reais, esta abordagem apresenta um desempenho eficiente (Taheri and Mammadov, 2013; Ray, 2019; Mahesh, 2020).

O algoritmo Naive Bayes utiliza o Teorema de Bayes para identificar a probabilidade de um conjunto de variáveis se enquadrar numa classe específica (Jadhav and Channe, 2016; Saravanan and Sujatha, 2018). A Equação 8 representa a expressão matemática aplicada no algoritmo.

$$P(C|X) = \frac{P(X|C) \times P(C)}{P(X)} \tag{8}$$

Onde:

- X = amostra de dados cuja classe não é conhecida.
- C = classe específica.
- $P(C|X)$ = probabilidade posterior da classe alvo.
- $P(X|C)$ = verosimilhança, probabilidade das variáveis X ocorrerem na classe C.
- $P(C)$ = probabilidade prévia da classe.
- $P(X)$ = probabilidade prévia das variáveis.

A Naive Bayes é uma metodologia de fácil implementação, apresenta um bom desempenho e não é afetado por caraterísticas irrelevantes. No entanto, perante conjuntos de dados complexos possui um desempenho inferior em comparação com outras técnicas de classificação (Jadhav and Channe, 2016; Ray, 2019).

2.4.4 Random Forest

A Random Forest é uma metodologia que pode ser aplicada a problemas de regressão, bem como problemas de classificação, apresentando uma estrutura única e versátil. Esta abordagem tornou-se uma ferramenta essencial para a análise de dados, demonstrando um desempenho superior na prática em comparação com outros métodos (Scornet, Biau and Vert, 2015; Genuer *et al.*, 2017).

A metodologia Random Forest combina a abordagem *bagging* e a seleção aleatória de caraterísticas para construir um conjunto de Árvores de Decisão. Com o *bagging*, cada Árvore de Decisão é elaborada utilizando uma amostra com substituição dos dados de treino. A Random Forest implementa uma montagem em paralelo, que combina vários classificadores de Árvores de Decisão em paralelo, como é apresentado na Figura 14, em diferentes subamostras do conjunto de dados e utiliza a votação por maioria ou o cálculo de médias para obter o resultado final. Desta forma, minimiza o *overfitting* e aumenta o controlo da previsão. O *overffiting* ocorre quando o algoritmo se adapta excessivamente aos dados de treino, resultando num modelo que não consegue fazer previsões precisas com outros dados (Fawagreh, Gaber and Elyan, 2014; Sarker, 2021).

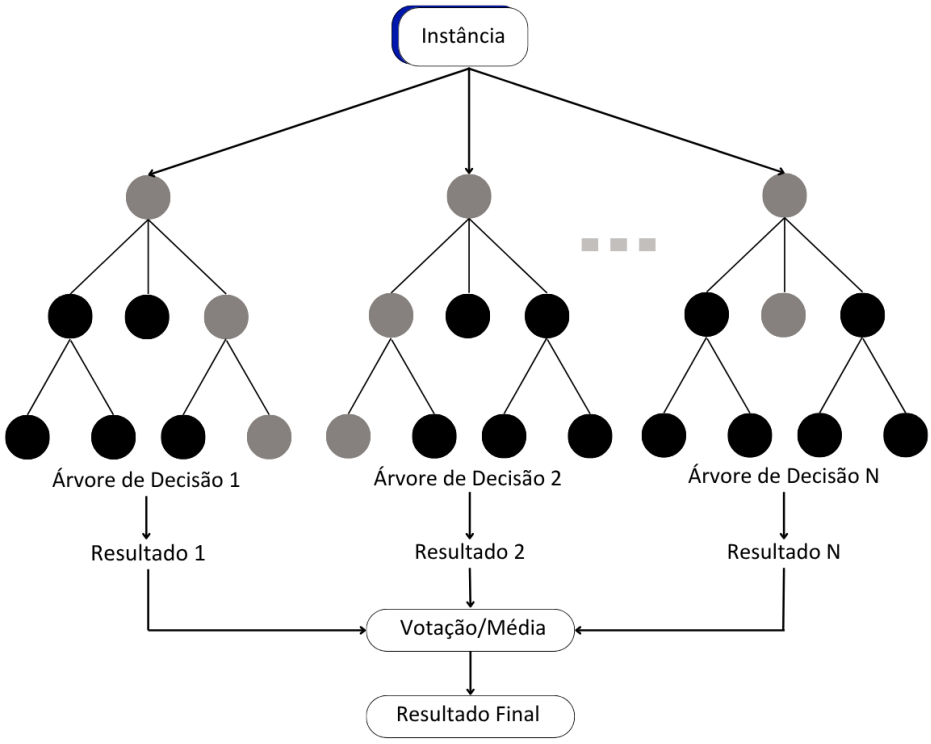


Figura 14 - Exemplo de Random Forest, baseado em Sarker (2021)

A Random Forest é reconhecida pela sua precisão e capacidade de lidar com estruturas de dados complexos. Esta técnica tem sido utilizada com sucesso em muitos problemas práticos, incluindo a previsão da qualidade do ar, reconhecimento de objetos 3D, bioinformática, entre outros (Scornet, Biau and Vert, 2015).

2.4.5 Regressão Linear

A Regressão Linear é uma das técnicas de Machine Learning mais populares, caraterizando-se por ser um algoritmo de regressão. A Regressão Linear tem como objetivo principal criar uma relação entre a variável dependente e uma ou mais variáveis independentes, sendo que a variável dependente é contínua e a(s) variável(is) independente(s) pode(m) ser contínua(s) ou discreta(s). Esta metodologia pode ser classificada como Regressão Linear simples ou Regressão Linear múltipla, definidas pelas Equações 9 e 10, respetivamente (Kavitha, Varuna and Ramya, 2017; Sarker, 2021).

$$y = a + bx + e \tag{9}$$

$$y = a + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n + e \tag{10}$$

Nas Equações 9 e 10, y representa a variável dependente, x indica a(s) variável(is) independente(s), a e b são constantes e e é o termo de erro. Estas equações podem ser utilizadas para prever o valor da variável dependente com base na(s) variável(is) independente(s) fornecida(s). A Regressão Linear simples é implementada apenas quando existe uma única variável independente, enquanto a Regressão Linear múltipla é utilizada quando existe mais do que uma variável independente no algoritmo (Kavitha, Varuna and Ramya, 2017; Sarker, 2021).

A Regressão Linear tem a vantagem de ser fácil de interpretar, porém, não é um método que deve ser implementado na maioria das aplicações práticas, pois simplifica demasiado os problemas do mundo real. Além disso, é uma metodologia que não deve ser aplicada a instâncias com padrões complexos (Ray, 2019).

2.4.6 Regressão Logística

A Regressão Logística é um modelo estatístico comum baseado em probabilidades utilizado para resolver problemas de classificação. Carateriza-se por utilizar uma função logística para determinar as probabilidades, também conhecida como função sigmoide. A Regressão Logística apresenta o resultado na forma binomial, uma vez que calcula a probabilidade de uma situação ocorrer ou não (representado por 0 e 1), com base nos valores das variáveis de *input* (Ray, 2019; Sarker, 2021). A expressão implementada na técnica de Regressão Logística é apresenta na Equação 11, sendo que Y representa a variável dependente, $X_1, \dots, X_p$ indica as variáveis independentes e $\beta_0, \beta_1, \dots, \beta_p$ são coeficientes do modelo (Couronné, Probst and Boulesteix, 2018).

$$P = \left(Y = 1 \middle| X_1, \dots, X_p\right) = \frac{\exp\left(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p\right)}{1 + \exp\left(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p\right)} \quad (11)$$

A Regressão Logística tem a vantagem de ser fácil de implementar e funciona bem com conjuntos de dados lineares. Contudo, apresenta a incapacidade de resolver problemas não lineares e as variáveis do modelo têm de ser independentes. A Regressão Logística pode ser utilizada para: previsão do risco de desenvolver uma determinada doença, diagnóstico do cancro, probabilidade de falha de um determinado processo, entre outros (Ray, 2019).

2.5 Trabalhos relacionados

O escalonamento de produção representa um papel crucial nos sistemas de fabrico, dado que, um esquema de produção eficaz é um fator fundamental que permite melhorar a eficiência da produção e a taxa da utilização de recursos. A resolução deste tipo de problemas exige a aplicação de algoritmos adequados. Destaca-se a utilização de Meta-heurísticas, visto que, podem ser facilmente modificadas de acordo com o problema em causa (Chen *et al.*, 2020; Agrawal *et al.*, 2021).

As Meta-heurísticas, embora sejam conhecidas como técnicas de aproximação ótimas para problemas de otimização complexos, apresentam dificuldades na sua implementação. Um dos principais desafios está relacionado com a seleção da Meta-heurística mais adequada para a resolução de um problema. Esta etapa é considerada complexa, uma vez que é difícil indicar uma Meta-heurística como a melhor de todas. Além disto, as Meta-heurísticas possuem parâmetros que necessitam de ser determinados, sendo que os seus valores influenciam o desempenho do modelo (Pereira, 2014; Pereira *et al.*, 2021).

Desta forma, técnicas de Machine Learning podem ser incorporadas às Meta-heurísticas com o intuito de melhor o desempenho do sistema de produção. Assim, considerando os tópicos abordados, foi realizado um estudo sobre trabalhos que abordam o problema de seleção de algoritmos e a afinação de parâmetros das Meta-heurísticas.

2.5.1 Seleção de algoritmos

A qualidade do escalonamento de produção pode ter um impacto significativo na produtividade do sistema global. Desta forma, a seleção do algoritmo de escalonamento torna-se um fator importante e, embora exista diferentes abordagens para qualquer problema, estas variam muito na qualidade das suas soluções. Assim, é fundamental escolher o algoritmo mais adequado para resolver uma instância concreta (Strassl and Musliu, 2022). Dado este problema, foram identificados dois artigos que se consideram relevantes para o tema desta dissertação, apresentados na Tabela 2. Ambos os artigos apresentam uma proposta de um modelo de seleção de algoritmos baseado em Machine Learning.

Tabela 2 - Trabalhos Relacionados de Seleção de Algoritmos

Autor	Algoritmo	Machine Learning	Tipo de Problema
Müller et al. (2022)	CPLEX; OR-Tools	Árvores de Decisão; Redes Neuronais Profundas	Job-Shop Flexível
Strassl and Musliu (2022)	Métodos Exatos; Métodos Heurísticos; Métodos Meta-heurísticos	K-Nearest Neighbors; Árvores de Decisão; Random Forest; Multilayer Perceptron	Job-Shop

O artigo de Müller et al. (2022) analisa o problema de seleção de algoritmos para o escalonamento de produção de Job-Shop Flexível. Os autores realizam um estudo comparativo entre cinco solucionadores considerando instâncias de teste retiradas da literatura, bem como instâncias de teste geradas aleatoriamente. Os resultados indicam que, quando se considera apenas a qualidade da solução, o solucionador CPLEX apresenta os melhores resultados, mas, quando se considera a capacidade de calcular rapidamente soluções ótimas, o solucionador OR-Tools apresenta resultados mais relevantes. Através desta análise, o estudo foca-se na elaboração de uma abordagem de seleção de algoritmos baseada em técnicas de Machine Learning para prever qual dos dois solucionadores, CPLEX ou OR-Tools, é o mais adequado para cada instância do problema.

As abordagens de Machine Learning utilizadas no artigo científico foram Árvores de Decisão e Redes Neuronais Profundas. Para evitar o *overffiting*, os autores pré-avaliaram os modelos no conjunto de validação. O desempenho de um modelo é medido em termos da sua precisão, ou seja, a percentagem de classificações corretas no conjunto de validação. Em relação às Árvores de Decisão, foram avaliadas sete variações desta técnica e, com base nos resultados, os autores decidiram selecionar as três melhores técnicas para avaliar o conjunto de teste. As abordagens que obtiveram os melhores resultados foram Random Forest, Logistic Model Tree e REP Tree, com valores de Accuracy de 86,36%, 85,36% e 84,25%, respetivamente. Relativamente à técnica de Redes Neuronais Profundas, foram selecionadas as técnicas Convolutional Neural Networks e Fully Connected Neural Network que demonstraram uma Accuracy de 84,1% e 86,28% no conjunto de validação, na respetiva ordem.

O estudo de Müller et al. (2022) apresentou uma definição de um conjunto de caraterísticas (*features*) das instâncias do problema. A definição do conjunto de caraterísticas é uma parte crucial de qualquer abordagem de seleção de algoritmos, uma vez que é utilizada para caraterizar as instâncias e depois relacionar estas características com o desempenho dos algoritmos. Os autores identificaram 151 caraterísticas que foram agrupadas em quatro categorias: tamanho da instância (28), flexibilidade (50), tempo de processamento (37) e regras de prioridade (36). Para além disso, os autores utilizaram uma razão de 80/20 para dividir o conjunto de instâncias (22984) em conjunto de treino e conjunto de validação, respetivamente.

Para a verificação final, os autores optaram por testar as abordagens de seleção de algoritmos num conjunto de teste separado composto por instâncias nunca vistas anteriormente.

Com base na análise final, é possível verificar que, no artigo de Müller et al. (2022), os modelos de seleção de algoritmos superam a utilização de um único solucionador quando restringidos a tempos de execução com valores relativamente pequenos. Os autores, para cada modelo, apresentam a informação sobre a percentagem de instâncias para as quais foi encontrada uma solução ótima com o solucionador selecionado dentro do limite de tempo, os rácios de qualidade médios, os tempos de execução médios e a pontuação final. Além disto, incluem informações sobre a percentagem da Accuracy. Os autores concluíram que, em média, a técnica de Árvores de Decisão baseada em Random Forest proporciona o melhor desempenho no modelo.

O artigo de Strassl and Musliu (2022) analisa o problema de seleção de algoritmos para o escalonamento de produção de Job-Shop. Os autores iniciam a sua investigação com uma análise ao espaço de instâncias, explorando tanto os *benchmarks* tradicionais quanto as novas instâncias geradas aleatoriamente. Com base nas instâncias existentes na literatura (242 instâncias), foi criado um conjunto de instâncias, das quais 3025 foram incluídas no processo de análise. Os autores optaram por elaborar as novas instâncias com o intuito de garantir uma representação mais abrangente do espaço de instâncias do problema de Job-Shop.

Para caraterizar as instâncias, o artigo apresenta um conjunto de caraterísticas (*features*), 604 no total. Os autores realçam que algumas caraterísticas não contêm informação ou estão fortemente correlacionadas com outras caraterísticas. No entanto, todas as caraterísticas são incluídas por uma questão de coerência.

Os algoritmos de solução considerados neste estudo podem ser divididos em três categorias distintas: Métodos Exatos, Métodos Heurísticos e Métodos Meta-heurísticos de Pesquisa Local. Cada uma destas categorias contém uma série de algoritmos que são utilizados no estudo. Os Métodos Exatos englobam CP Optimizer, OR-Tools, Chuffed e CPLEX e, os Métodos Heurísticos incluem Dispatching rule-based Heuristics e Shifting Bottleneck Heuristic. Os Métodos Meta-heurísticos de Pesquisa Local são compostos por Random-restart Hill Climbing, Tabu Search e Simulated Annealing.

Os autores realizaram um estudo comparativo entre os algoritmos de solução referidos anteriormente. A partir desta análise básica, os algoritmos que apresentam o melhor desempenho são o CP Optimizer, Tabu Search e OR-Tools, sendo que, cada um, tem o seu próprio subconjunto distinto de instâncias nas quais se destacam. O CP Optimizer obteve o melhor desempenho e o pior caso, enquanto o algoritmo Tabu Search se destacou em relação ao número de instâncias para as quais obteve a melhor solução.

No artigo de Strassl and Musliu (2022), para a elaboração dos modelos de seleção de algoritmos, as técnicas de Machine Learning implementadas foram K-Nearest Neighbors, Árvores de Decisão, Random Forest e Multilayer Perceptron. Os autores utilizam as métricas P_6 e

Coeficiente de Correlação de Matthews (MCC) como medidas principais na análise do desempenho dos modelos. A métrica P_6 representa a razão entre o *makespan* obtido pelo algoritmo escolhido pelo modelo (C_{max}) e o *makespan* do melhor algoritmo para a mesma instância (C_{max}^{best}). Neste estudo, é aplicada uma razão de 50/50 para dividir o conjunto de instâncias em conjunto de treino e conjunto de teste.

Com base na análise final, no artigo de Strassl and Musliu (2022), é possível verificar que a previsão do melhor algoritmo para qualquer instância utilizando modelos de Machine Learning resultou numa melhoria estatisticamente significativa do desempenho em relação ao desempenho de qualquer algoritmo individual. Os autores concluíram que a abordagem de Machine Learning baseada em Random Forest proporciona o melhor desempenho no modelo. A aplicação de Random Forest no modelo permitiu prever o algoritmo mais adequado para 90% das instâncias e alcançou uma média de P_6 de 1,0016. O melhor resultado do modelo é seguido pelas técnicas Multilayer Perceptron e Árvores de Decisão, apresentando diferenças mínimas.

Após a análise dos artigos científicos de Müller et al. (2022) e Strassl and Musliu (2022), é possível identificar que, ambos os estudos, apresentam semelhanças entre si. Verifica-se que, apesar da incorporação de metodologias diferentes e a utilização de diferentes indicadores de desempenho, os modelos de seleção de algoritmos baseados em Machine Learning apresentam um desempenho superior quando comparados com os algoritmos individuais. É de realçar que, para ambos os estudos, o sistema de seleção de algoritmos baseado na abordagem de Random Forest é o modelo que alcança os melhores resultados. Apesar da identificação de algumas tendências, esta dissertação concentrar-se-á em algoritmos de solução baseados em técnicas de Meta-heurísticas, enquanto os modelos de seleção de algoritmos baseados em Machine Learning englobarão abordagens de aprendizagem supervisionada.

2.5.2 Afinação de parâmetros

As Meta-heurísticas podem ser combinadas com técnicas de Machine Learning para melhorar o desempenho do sistema de produção. As abordagens de Machine Learning são integradas com o objetivo de auxiliar na parametrização das Meta-heurísticas, visto que, a parametrização tem um forte impacto no desempenho ou eficácia da Meta-heurística (Huang, Li and Yao, 2020; Karimi-Mamaghan *et al.*, 2022; Santos, Madureira and Varela, 2022).

Entre os diversos artigos dedicados à aplicação de técnicas de Machine Learning para parametrizar Meta-heurísticas, é possível identificar diferentes caraterísticas que os distinguem, como o tipo de parametrização e as técnicas de Machine Learning e Meta-heurísticas utilizadas na resolução do determinado problema.

Na Tabela 3 é possível observar a identificação de sete artigos que se destacam pela sua semelhança ao tema desta dissertação. Verifica-se que grande parte dos estudos realizam uma parametrização *online*, na qual os parâmetros são ajustados dinamicamente durante a execução do algoritmo. Dentro desse contexto, a abordagem de Machine Learning mais utilizada para a parametrização das Meta-heurísticas é a aprendizagem por reforço. A

aprendizagem por reforço é uma metodologia que possibilita a um agente aprender, utilizando informações obtidas a partir das suas ações e experiências, com o objetivo de maximizar uma recompensa numéria, aumentando o desempenho do modelo. Além disto, é possível observar que a Meta-heurística mais utilizada para a resolução dos problemas identificados é o Algoritmo Genético, destacando-se por ser a abordagem mais comum para resolver problemas práticos de otimização combinatória.

Tabela 3 - Trabalhos Relacionados de Afinação de Parâmetros

Autor	**Meta-heurística**	**Machine Learning**	**Tipo de Parametrização**	**Tipo de Problema**
Tatsis and Parsopoulos (2020)	Differential Evolution	Aprendizagem por Reforço (Algoritmo Reinforce)	Online	Problema de otimização contínua em grande escala
Chen et al. (2020)	Algoritmo Genético	Aprendizagem por Reforço	Online	Job-Shop Flexível
Huynh, Do and Lee (2021)	Differential Evolution	Aprendizagem por Reforço (Q-learning)	Online	Problema de otimização estrutural
Tondut et al. (2022)	Particle Swarm Optimization; Differential Evolution; Clonal Selection Algorithm	Metodologia Kriging	Offline	12 Funções matemáticas
Pikalov and Pismerov (2023)	Algoritmo Genético	Feedforward Neural Network	Online	Problema W-model
Reijnen et al. (2023)	Algoritmo Genético	Proximal Policy Optimization	Online	Job-Shop Flexível
Yang et al. (2025)	Algoritmo Genético	Aprendizagem por Reforço (Q-learning)	Online	Job-Shop Flexível com transferência

Nos sete artigos identificados, os modelos propostos e desenvolvidos pelos autores foram comparados com outras técnicas implementadas na resolução do respetivo problema, sendo que todos os autores identificaram vantagens no sistema, nomeadamente um aumento geral do desempenho do modelo. Apesar das tendências identificadas nos estudos, nesta dissertação, a incorporação de Machine Learning para parametrização das Meta-heurísticas será aplicada a problemas de escalonamento de Job-Shop, utilizando técnicas de aprendizagem supervisionada, com a realização de uma parametrização *offline*.

Ao observar a Tabela 3, verifica-se que apenas um artigo implementa a parametrização *offline* e, como esta dissertação realizará este tipo de parametrização, é fundamental analisar este artigo científico. O artigo de Tondut et al. (2022) foi elaborado com o objetivo de desenvolver

uma metodologia automatizada para afinar os parâmetros de algoritmos Meta-heurísticos para problemas de otimização. A combinação ótima dos parâmetros é determinada por um processo de *kriging*, permitindo que a metodologia de afinação automática de parâmetros de Meta-heurísticas forneça a configuração ótima do algoritmo para cada problema, de modo a alcançar uma melhor previsão de convergência.

O modelo proposto é utilizado para afinar três algoritmos Meta-heurísticos diferentes, Particle Swarm Optimization, Differential Evolution e Clonal Selection. Cada um destes algoritmos é aplicado a doze funções matemáticas, sendo cinco unimodais e sete multimodais. Os autores, para analisar a precisão de convergência do método proposto, comparam o modelo com as configurações clássicas habitualmente utilizadas na literatura.

Para avaliar a eficiência do modelo proposto, os autores realizam 20 cálculos de otimização nas dimensões 10, 30 e 50 em cada uma das funções matemáticas, sendo calculados a média e o desvio padrão. Para além disto, o número de interações no algoritmo Meta-heurístico é limitado a 500 interações.

Através da análise final dos resultados, os autores verificaram uma melhoria média da precisão de convergência em relação aos valores padrões de 62,02%, 69,12% e 64,94% na dimensão 10, 30 e 50, respetivamente. Para além disto, a metodologia proposta mostra uma melhoria significativa no desempenho dos algoritmos em 97,2% dos problemas aplicados.

2.6 Sumário

Neste capítulo foram descritos um conjunto de temas que são abordados nesta dissertação, nomeadamente os problemas de escalonamento, as Meta-heurísticas e a Machine Learning. Adicionalmente, foi incluída uma subsecção dedicada à revisão de estudos relacionados, com destaque para os trabalhos que abordam a seleção de algoritmos para resolução de problemas de escalonamento e a afinação de parâmetros das Meta-heurísticas.

Na subsecção dos problemas de escalonamento foi essencial a caracterização das duas técnicas de escalonamento mais estudadas, o Flow-Shop e o Job-Shop. Na subsecção das Meta-heurísticas foram apresentados alguns algoritmos (Simulated Annealing, Tabu Search, Algoritmos Genéticos, Ant Colony Optimization, Particle Swarm Optimization, Artificial Bee Colony), juntamente com uma análise aprofundada sobre o problema de afinação de parâmetros. Por último, foram abordadas as diferentes classificações da metodologia de Machine Learning (aprendizagem supervisionada, aprendizagem não supervisionada, aprendizagem semi-supervisionada e aprendizagem por reforço), sendo destacados alguns algoritmos da aprendizagem supervisionada (Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest, Regressão Linear, Regressão Logística).

3 Metodologia

Neste capítulo é elaborado uma descrição de como foi implementada a metodologia de CRISP-DM nesta dissertação. Para além disso, é realizada a identificação e contextualização do problema que incentiva a elaboração desta dissertação. Adicionalmente, é desenvolvida uma análise dos dados brutos utilizados neste projeto.

3.1 Aplicação do CRISP-DM

Para a elaboração desta dissertação foi essencial a implementação do modelo CRISP-DM. Esta metodologia permite, de forma organizada, descrever as etapas realizadas neste projeto, desde a compreensão do problema até à implementação.

Como referido na secção 1.3, o CRISP-DM tem início na fase de compreensão do problema, destinada para definir o contexto de escalonamento de produção e identificar as técnicas de Meta-heurísticas e Machine Learning adotadas no estudo. A etapa seguinte corresponde à compreensão dos dados, na qual é realizada a recolha dos dados brutos e é elaborada uma análise exploratória dos dados, com o objetivo de identificar padrões relevantes. Estas duas fases inicias são descritas nas subsecções seguintes.

A terceira etapa do modelo CRISP-DM corresponde à preparação dos dados. Contudo, dado que o objetivo desta dissertação é a elaboração de um sistema de seleção e autoparametrização das Meta-heurísticas, foi essencial dividir o estudo em duas partes. A primeira parte destina-se à construção do modelo preditivo para a seleção da Meta-heurística mais adequada para a resolução do problema de escalonamento. A segunda parte é orientada para a elaboração do modelo preditivo para a afinação dos parâmetros da Meta-heurística selecionada. Com esta divisão, as etapas de preparação de dados, modelação e avaliação foram realizadas duas vezes, uma para cada modelo. Estas fases encontram-se descritas nos capítulos 4 e 5. Na Figura 15 é possível observar um esquema geral da implementação de CRISP-DM nesta dissertação.

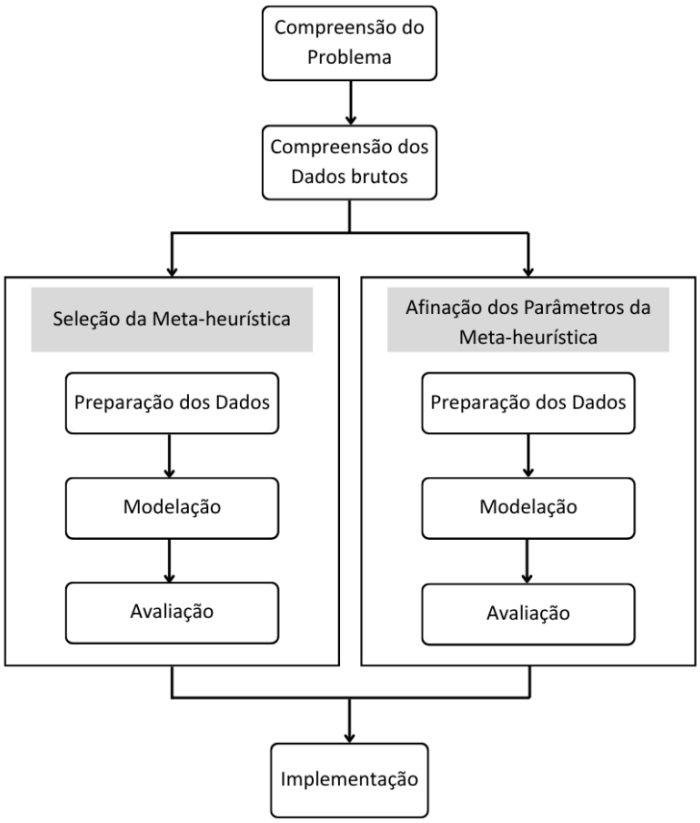


Figura 15 - Esquema geral do desenvolvimento da metodologia CRISP-DM

A última etapa do CRISP-DM corresponde à implementação, porém, esta fase não foi realizada devido a limitações de tempo. No entanto, o sistema desenvolvido nesta dissertação poderá ser utilizado como base para a elaboração de um sistema autónomo que seleciona e realiza a afinação dos parâmetros da Meta-heurística para novas instâncias de escalonamento de produção. Esta proposta está descrita com maior detalhe na secção 6.2.

Todas as bases de dados e códigos desenvolvidos nesta dissertação estão disponíveis publicamente no Github: https://github.com/anabcsilva-20/selecao-autoparametrizacao-metaheuristicas.

3.2 Compreensão do problema

O escalonamento de produção desempenha um papel crucial nos sistemas de fabrico, revelando-se como um fator que influencia a eficiência operacional das empresas. O objetivo principal é definir um plano que atinja as metas de produção, assegurando que todas as restrições sejam cumpridas e os objetivos otimizados. A resolução de problemas de escalonamento exige a implementação de algoritmos adequados, destacando-se a utilização de Meta-heurísticas, metodologias capazes de fornecer soluções satisfatórias em tempos de execução razoáveis.

As Meta-heurísticas, apesar de demonstrarem a capacidade de poderem ser aplicadas em diversos problemas, apresentam dificuldades na sua implementação. Um dos principais desafios é identificar qual a Meta-heurística mais adequada para a resolução de um problema. Esta etapa é considerada frequentemente complexa, visto que requer uma análise detalhada do tipo de problema e o conhecimento profundo das técnicas disponíveis. Do mesmo modo, é difícil eleger uma Meta-heurística como a melhor de todas, uma vez que cada abordagem possui vantagens e desvantagens.

Além disso, as Meta-heurísticas possuem parâmetros que necessitam de ser corretamente definidos. Devido ao controlo que os parâmetros têm no comportamento das Meta-heurísticas, o processo de parametrização influencia a eficácia e o desempenho desta abordagem. Esta etapa não é uma tarefa simples, sendo necessário a ajuste dos parâmetros sempre que o modelo lida com novas instâncias.

As Meta-heurísticas podem ser combinadas com técnicas de Machine Learning para melhorar o desempenho do sistema de produção. As abordagens de Machine Learning podem ser integradas com o objetivo de auxiliar na seleção e parametrização das Meta-heurísticas, uma vez que estas etapas têm um grande impacto no desempenho e eficácia do sistema.

Neste contexto, esta dissertação dedica-se a oferecer uma contribuição para a resolução de problemas de escalonamento de produção. Neste sentido, é proposto um sistema de seleção e autoparametrização de Meta-heurísticas que utiliza Machine Learning para otimizar o escalonamento de tarefas. O modelo em questão abrange unicamente problemas de escalonamento de produção do tipo Job-Shop e a parametrização das Meta-heurísticas é realizada de forma *offline*. O sistema considera apenas a utilização de cinco Meta-heurísticas (Tabu Search, Simulated Annealing, Algoritmos Genéticos, Ant Colony Optimization, Particle Swarm Optimization) e recorre exclusivamente a técnicas de aprendizagem supervisionada de Machine Learning (Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest, Regressão Linear, Regressão Logística).

3.3 Compreensão dos dados

A base de dados utilizada para a elaboração desta dissertação provém da tese de doutoramento de Pereira (2014). Pereira (2014), no seu trabalho, teve como objetivo geral a criação de um sistema inteligente para escalonamento assistido por aprendizagem.

A base de dados bruta é composta por 3116 registos (linhas), contendo 39 variáveis (colunas). Cada registo representa uma observação individual, enquanto cada coluna representa uma variável associada a essas observações. A Tabela 4 demonstra uma caraterização detalhada de algumas variáveis presentes na base de dados. Cada variável é classificada de acordo com as suas caraterísticas (qualitativa ou quantitativa) e é apresentada uma pequena descrição da mesma.

Tabela 4 - Classificação das variáveis da base de dados

Variável	Tipo de variável	Descrição
MH	Qualitativa Nominal	Nome da Meta-heurística a utilizar (TS, SA, GA, ACO, PSO)
NJobs	Quantitativa Discreta	Número de tarefas da instância
NMachines	Quantitativa Discreta	Número de máquinas da instância
ProbType	Qualitativa Nominal	Tipo de problema (Flow-Shop, Job-Shop)
MultiLevel	Qualitativa Binária	Indica se a instância corresponde ou não a um problema multinível
CmaxOpt	Quantitativa Discreta	Valor ótimo de makespan (C_{max}) conhecido
InitSol	Qualitativa Nominal	Heurística construtiva utilizada para construir a solução a solução inicial (sequência de nível)
ObjFunc	Qualitativa Nominal	Função objetivo usada para a minimização do problema (C_{max}, WT)
Cmax	Quantitativa Discreta	Valor de makespan (C_{max}) obtido
TimeExec	Quantitativa Contínua	Tempo de execução (em segundos)

A Tabela 4 não demonstra todas as variáveis presentes na base de dados. As restantes variáveis correspondem aos parâmetros específicos de cada Meta-heurística considerada no desenvolvimento desta dissertação. Tratam-se de variáveis quantitativas, cujo número varia consoante a abordagem adotada. Neste caso, foram consideradas 4 variáveis para a Meta-heurística Tabu Search, 4 para a Simulated Annealing, 5 para os Algoritmos Genéticos, 6 para o Ant Colony Optimization e 10 para o Particle Swarm Optimization.

Na base de dados, uma instância de um problema de escalonamento é caraterizada pelo número de tarefas, número de máquinas, pelo tipo de problema, pelo facto de existirem relações multinível e pelo seu valor ótimo de *makespan*. Importa salientar que apenas é possível encontrar problemas de escalonamento (variável 'ProbType') do tipo Job-Shop e não existe relações multinível (variável 'MultiLevel' com valor de 0).

A base de dados é constituída por 80 instâncias distintas, cada uma contendo diversos registos correspondentes aos valores de *makespan* (C_{max}), obtidos a partir da aplicação das cinco Meta-heurísticas consideradas. A Tabela 5 demonstra a distribuição de cada Meta-heurísticas nas 80 instâncias.

Tabela 5 - Presença das Meta-heurísticas nas instâncias de escalonamento

Meta-heurística	Número de instâncias	Percentagem (%)
Simulated Annealing	80	100
Tabu Search	79	98,75
Particle Swarm Optimization	79	98,75
Algoritmos Genéticos	64	80
Ant Colony Optimization	28	35

Verifica-se que três Meta-heurísticas apresentam valores altos de representatividade nas instâncias de escalonamento. A metodologia Simulated Annealing exibe uma utilização constante, indicando uma presença de 100%, e as Meta-heurísticas Tabu Search e Particle Swarm Optimization revelam um valor de 98,75% nas instâncias. Relativamente à metodologia Algoritmos Genéticos, esta exibe um valor de 80%, revelando uma aplicação significativa, apesar de ser menos consistente comparando com as três abordagens anteriores. No entanto, em contraste, a Meta-heurística Ant Colony Optimization demonstra uma representatividade bastante baixa, estando presente em apenas 28 instâncias, obtendo um valor de 35%.

Esta análise sugere uma preferência de utilização das Meta-heurísticas Simulated Annealing, Tabu Search e Particle Swarm Optimization, possivelmente associada a uma adequação maior destas técnicas aos problemas de escalonamento avaliados. Contudo, é importante destacar que, em algumas instâncias, é possível observar vários registos da mesma Meta-heurística, resultante da utilização de diferentes parâmetros. Como tal, é essencial a elaboração de um estudo da distribuição das Meta-heurísticas pelos registos da base de dados (Figura 16).

O Gráfico de barras (Figura 16) indica a frequência absoluta de registos associado a cada Meta-heurística. Nota-se que a Meta-heurística Simulated Annealing destaca-se em relação às outras, apresentando o maior número de ocorrências, um total de 1185 registos. As metodologias Tabu Search e Particle Swarm Optimization exibem um total de 884 e 716 registos, demonstrando uma presença significativa, embora inferior à da técnica Simulated Annealing. Em relação à abordagem de Algoritmos Genéticos, esta demonstra uma participação baixa, revelando apenas 270 registos. No entanto, a Meta-heurística Ant Colony Optimization é a que apresenta o número mais baixo de registos, nomeadamente 61 ocorrências.

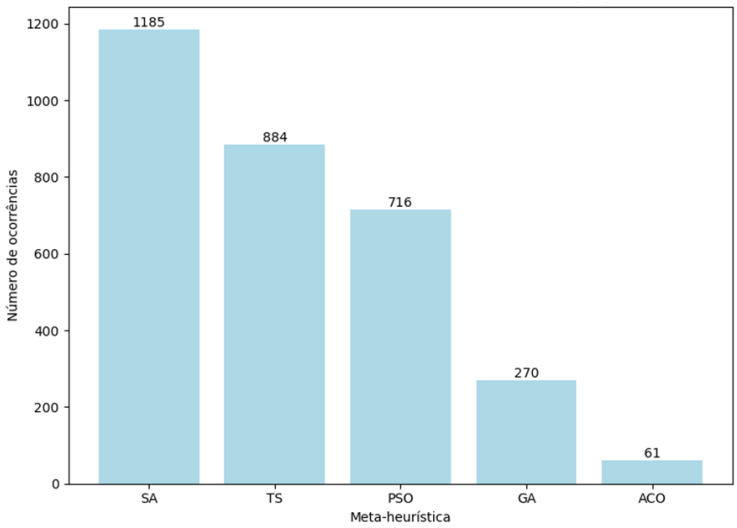


Figura 16 - Número de ocorrências de cada Meta-heurística

Através da análise da Figura 16, é possível observar uma falta de balanceamento em relação à utilização das Meta-heurísticas. Isto significa que, algumas metodologias foram utilizadas de forma diversificada, demonstrando um valor alto de ocorrências, enquanto outras foram implementadas de modo mais específico, exibindo um número baixo de registos.

Na base de dados, torna-se relevante a elaboração de uma análise à dispersão dos valores de *makespan* obtidos (C_{max}) nas instâncias. Para tal, procedeu-se à realização de gráficos boxplot (Anexo A) para cada instância de escalonamento, tendo sido identificados 65 boxplots que apresentam *outliers*, valores atípicos. A presença frequente de *outliers* indica a existência de uma variação significativa entre os valores de *makespan* obtidos pelas Meta-heurísticas. Importa destacar que, na maioria dos boxplots, existe uma concentração de *outliers* à direita, que correspondem a valores de *makespan* significativamente superiores. Esta tendência reforça a ideia de que o desempenho das Meta-heurísticas é afetada pelas caraterísticas do problema. Adicionalmente, a presença de *outliers* pode ser justificada pela configuração dos parâmetros das Meta-heurísticas que influencia o desempenho e a eficácia da abordagem na instância.

Apesar da identificação das 65 instâncias que contêm *outliers*, também é possível observar a presença de 15 boxplots que não contêm valores atípicos. Isto indica que, diferentes Meta-heurísticas podem, às vezes, obter desempenhos semelhantes numa determinada instância de escalonamento. No Anexo A é possível observar os dos gráficos boxplot das instâncias que contêm *outliers*.

Para concluir a análise dos dados brutos, efetuou-se o cálculo do *gap* percentual das cinco Meta-heurísticas. O *gap* percentual é uma medida que realiza uma comparação entre o resultado obtido de *makespan* e o valor ótimo de *makespan*. Por outras palavras, mostra a diferença relativa entre o valor obtido e o resultado ideal, em percentagem. Esta medida é calculada da seguinte forma:

$$Gap\ Percentual = \frac{(C_{max} - C_{max}Opt)}{C_{max}Opt} \times 100 \quad (12)$$

A Tabela 6 apresenta o valor do *gap* percentual médio, mínimo e máximo de cada Meta-heurística. Verifica-se que, a técnica Tabu Search foi, em média, a metodologia que obteve soluções mais próximas do ótimo, apresentando um *gap* percentual médio de 44,76%. No entanto, em contraste, a Meta-heurística Ant Colony Optimization obteve o maior valor de *gap* percentual médio, de 52,97%, indicando que as soluções atingidas não são próximas do valor ótimo, demonstrando um desempenho inferior em relação às outras abordagens. Em relação as Meta-heurísticas Simulated Annealing, Particle Swarm Optimization e Algoritmos Genéticos, estas apresentam valores de *gap* percentual médios relativamente próximos, de 48,63%, 49,47% e 45,81%, respetivamente.

Tabela 6 - Gap percentual médio, mínimo e máximo das Meta-heurísticas

Meta-heurística	Gap Percentual Médio	Gap Percentual Mínimo	Gap Percentual Máximo
Simulated Annealing	48,63	0	222,06
Tabu Search	44,76	0	126,89
Particle Swarm Optimization	49,47	0	252,56
Algoritmos Genéticos	45,81	0	139,03
Ant Colony Optimization	52,97	6,24	149,64

Através da análise da Tabela 6, verifica-se que as metodologias Simulated Annealing, Tabu Search, Particle Swarm Optimization e Algoritmos Genéticos possuem um *gap* percentual mínimo de 0, o que indica que, pelo menos numa instância de escalonamento, estas técnicas conseguiram obter um valor de *makespan* igual ao valor ótimo. Analisando os valores de *gap* percentual máximos, percebe-se que a Particle Swarm Optimization obteve a maior diferença entre o valor de obtido e o número ótimo, de 252,56%. Por outro lado, a metodologia Tabu Search é a que apresenta o *gap* percentual máximo mais baixo, de 126,89%.

3.4 Sumário

Neste capítulo é apresentado, de forma detalhada, o desenvolvimento da metodologia CRISP-DM no âmbito desta dissertação. Esta explicação permite uma melhor compreensão da estrutura e do processo de elaboração do projeto, uma vez que a construção do sistema de seleção e autoparametrização das Meta-heurísticas foi divido em duas etapas. Além disso, é apresentado uma compreensão do problema, com o objetivo de contextualizar a identificar o problema que incentivou a realização desta dissertação. Para complementar, é realizada uma

análise exploratória dos dados brutos utilizados no projeto. Nesta secção são identificadas as variáveis que compõem a base de dados e são apresentadas algumas análises preliminares realizadas com o objetivo de compreender a dispersão e a variabilidade dos dados.

4 Modelo preditivo – Seleção da Meta-heurística

O presente capítulo tem como objetivo descrever as principais etapas envolvidas na elaboração e avaliação do modelo preditivo desenvolvido para identificar a Meta-heurística mais adequada para a resolução de um problema de escalonamento. Em primeiro lugar, é abordado o processo de preparação dos dados, onde os dados brutos são transformados num formato adequado para a análise. Na etapa seguinte, é apresentada a fase da modelação, na qual são selecionadas as técnicas mais adequada para o problema. Por fim, procede-se à avaliação dos resultados dos indicadores de desempenho.

4.1 Preparação dos dados

Durante as análises realizadas aos dados brutos, verificou-se que a Meta-heurística Ant Colony Optimization apresenta uma representatividade bastante reduzida no conjunto de dados. Em particular, os dados relativos a esta abordagem correspondem apenas a uma pequena parte do total das observações, com 61 registos. A escassa ocorrência da Meta-heurística pode afetar a eficácia e qualidade do modelo preditivo, nomeadamente nos algoritmos de Machine Learning.

Ainda, através da análise estatística e exploratória, foi possível observar que a Meta-heurística exibe um *gap* percentual médio elevado (52,97%), demonstrando um desempenho inferior comparando com as restantes Meta-heurísticas. Adicionalmente, através da análise do *gap* percentual mínimo verifica-se que esta abordagem, em nenhum momento, conseguiu obter um valor igual ao número ótimo (*gap* percentual mínimo=6,24%). Posto isto, procedeu-se à remoção dos registos da Meta-heurística Ant Colony Optimization, de forma a assegurar a robustez e qualidade dos modelos preditivos.

A próxima transformação da base de dados teve como intuito reduzir o número de registos, considerando, por instâncias, apenas as ocorrências que obtiveram melhores resultados. Para

cada instância de escalonamento, os valores de *makespan* obtidos (variável 'Cmax') foram ordenados por ordem crescente, considerando-se apenas metade dos registos de cada instância. A Figura 17 apresenta um exemplo da realização deste processo, estando assinalado a amarelo os registos que serão incluídos na base de dados.

MH	NJOBS	NMACHINES	CMAXOPT	CMAX	TIMEEXEC
GA	50	10	2972	5094	429,15
SA	50	10	2972	5191	7,55
GA	50	10	2972	5205	420,7
TS	50	10	2972	5209	85,38
GA	50	10	2972	5223	289,1
SA	50	10	2972	5237	7,21
PSO	50	10	2972	5246	83,41
TS	50	10	2972	5282	53,87
GA	50	10	2972	5298	422,69
GA	50	10	2972	5301	302,35
GA	50	10	2972	5345	399,36
TS	50	10	2972	5347	54,07
PSO	50	10	2972	5353	73,76
GA	50	10	2972	5353	439,51
GA	50	10	2972	5397	330,01
GA	50	10	2972	5418	445,13
TS	50	10	2972	5442	53,68
TS	50	10	2972	5455	53,85
GA	50	10	2972	5464	442,56
PSO	50	10	2972	5464	81,72
GA	50	10	2972	5488	454,66
GA	50	10	2972	5508	434,02
TS	50	10	2972	5522	44,5
TS	50	10	2972	5535	83,97
GA	50	10	2972	5578	458,36
TS	50	10	2972	5613	84,34
TS	50	10	2972	5772	102,88
TS	50	10	2972	5781	60,27
PSO	50	10	2972	5928	83,77
SA	50	10	2972	6448	2,6
PSO	50	10	2972	6626	0,4
SA	50	10	2972	6688	1,73
SA	50	10	2972	6908	1,57
PSO	50	10	2972	6981	0,26
PSO	50	10	2972	7108	149,11
SA	50	10	2972	7869	5,97
SA	50	10	2972	7933	8,67
SA	50	10	2972	9459	6,23

Figura 17 - Excerto de uma instância de escalonamento - Preparação dos dados: seleção de metade dos registos

Observando a Figura 17, é possível verificar que a instância contém um total de 38 registos e apenas 19 ocorrências é que são selecionadas para a base de dados. Contudo, verifica-se que o último registo com um valor de *makespan* de 5464 não é o único, a ocorrência seguinte apresenta o mesmo valor de *makespan*. Para determinar qual dos registos é que deve ser selecionado, realiza-se uma análise das Meta-heurísticas de cada registo. Caso apenas uma das Meta-heurísticas se encontre representada na base de dados da instância (parte amarela), esta ocorrência é selecionada. Contudo, considerando o exemplo apresentado, como ambas as Meta-heurísticas Algoritmos Genéticos e Particle Swarm Optimization já estão incluídas no conjunto de dados, a escolha é sustentada pelo tempo de execução (variável 'TimeExec'). O registo que apresentar um menor tempo de execução é selecionado, sendo considerada a ocorrência associada à Meta-heurística Particle Swarm Optimization.

A implementação destas duas transformações reduziu a base de dados para 1535 registos, mas, ainda assim, surgiu a necessidade da realização de mais uma modificação, de modo a aumentar a eficiência dos dados. Procedeu-se à identificação do menor valor de *makespan* (C_{max}) por instância de escalonamento. Caso esse valor estiver associado a mais do que uma Meta-heurística, todos os registos dessas abordagens são incorporados na base de dados e o processo é interrompido, procedendo-se à análise da instância seguinte (Figura 18). Caso exista apenas uma Meta-heurística associada, os seus registos são considerados na base de dados. Em seguida, realiza-se a média dos valores de *makespan* das outras Meta-heurísticas presentes no conjunto

de dados inicial. A abordagem que apresentar a menor média terá os seus registos incorporados na base de dados (Figura 19).

MH	NJOBS	NMACHINES	CMAXOPT	CMAX
PSO	10	10	930	1305
SA	10	10	930	1305
TS	10	10	930	1306
SA	10	10	930	1309
TS	10	10	930	1310
SA	10	10	930	1312
SA	10	10	930	1320
TS	10	10	930	1320
TS	10	10	930	1322
PSO	10	10	930	1344
TS	10	10	930	1354
SA	10	10	930	1360
TS	10	10	930	1373
TS	10	10	930	1373
TS	10	10	930	1373
SA	10	10	930	1378
SA	10	10	930	1378
TS	10	10	930	1391

Menor valor. Registo de duas Meta-heurísticas - incorporação na base de dados.

Base de dados final:				
MH	NJOBS	NMACHINES	CMAXOPT	CMAX
PSO	10	10	930	1305
SA	10	10	930	1305
SA	10	10	930	1309
SA	10	10	930	1312
SA	10	10	930	1320
PSO	10	10	930	1344
SA	10	10	930	1360
SA	10	10	930	1378
SA	10	10	930	1378

Figura 18 - Excerto de uma instância escalonamento - Preparação de dados: várias Meta-heurísticas com o menor valor de *makespan*

MH	NJOBS	NMACHINES	CMAXOPT	CMAX
GA	50	10	2972	5094
SA	50	10	2972	5191
GA	50	10	2972	5205
TS	50	10	2972	5209
GA	50	10	2972	5223
SA	50	10	2972	5237
PSO	50	10	2972	5246
TS	50	10	2972	5282
GA	50	10	2972	5298
GA	50	10	2972	5301
GA	50	10	2972	5345
TS	50	10	2972	5347
PSO	50	10	2972	5353
GA	50	10	2972	5353
GA	50	10	2972	5397
GA	50	10	2972	5418
TS	50	10	2972	5442
TS	50	10	2972	5455
PSO	50	10	2972	5464

Menor valor. Os registos da Meta-heurística Algoritmos Genéticos será incorporada na base de dados.

Cálculo médias:	
SA	5214
TS	5347
PSO	5354,33

Meta-heurística Simulated Annealing com menor valor de média.

Base de dados final:				
MH	NJOBS	NMACHINES	CMAXOPT	CMAX
GA	50	10	2972	5094
SA	50	10	2972	5191
GA	50	10	2972	5205
GA	50	10	2972	5223
SA	50	10	2972	5237
GA	50	10	2972	5298
GA	50	10	2972	5301
GA	50	10	2972	5345
GA	50	10	2972	5353
GA	50	10	2972	5397
GA	50	10	2972	5418

Figura 19 - Excerto de uma instância de escalonamento - Preparação de dados: uma Meta-heurística com menor valor de *makespan*

Com a implementação de todas as transformações, a base de dados final ficou com 1008 registos no total.

4.2 Modelação

O modelo foi elaborado recorrendo à aplicação Visual Studio Code, com a implementação das ferramentas Python 3.11.4 e Jupyter Notebook. O modelo preditivo desenvolvido é um algoritmo de classificação, projetado para determinar a Meta-heurística mais adequada para a

resolução de instâncias de problemas de escalonamento. Desta forma, foram implementadas cinco técnicas de Machine Learning, nomeadamente: Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest e Regressão Logística. Na Figura 20 é possível observar o esquema geral utilizado na elaboração deste modelo preditivo.

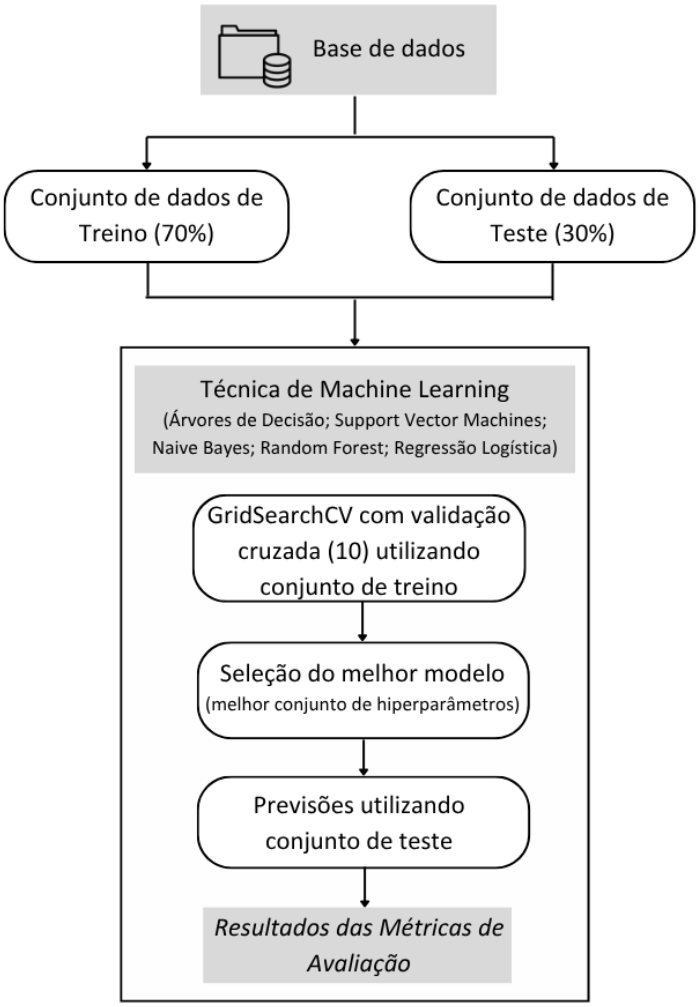


Figura 20 - Esquema geral do modelo preditivo de seleção das Meta-heurísticas

Neste modelo, a variável dependente considerada foi a variável 'MH', enquanto as variáveis independentes selecionadas para análise foram 'NJobs', 'NMachines', 'ObjFunc' e 'CmaxOpt'. Os dados foram divididos de forma estratificada em conjunto de treino e conjunto de teste numa proporção de 70/30, respetivamente. A divisão estratificada divide os dados em conjunto de treino e teste garantindo que as proporções das classes da variável dependente sejam mantidas em ambos os conjuntos. Esta técnica de divisão tornou-se benéfica, uma vez que a base de dados não se encontra equilibrada.

Na elaboração do modelo preditivo, para a otimização dos hiperparâmetros das abordagens de Machine Learning, foi implementado a técnica GridSearchCV. Esta estratégia tem como intuito selecionar a configuração que maximiza o desempenho do modelo através de uma análise das combinações possíveis dos parâmetros fornecidos. O elemento de validação cruzada também

foi implementado com um número de 10 *folds*, o que significa que, os dados são divididos em 10 partes, considerando que, em cada iteração, uma parte é utilizada para teste e as restantes para treino. A métrica de avaliação utilizada para identificar o melhor conjunto de hiperparâmetros foi o F1 Score ponderado (f1_weighted). Esta medida considera o desequilíbrio entre as classes evitando que a avaliação do modelo seja influenciada pelas classes que apresentam um maior número de registos. Na Tabela 7 é possível observar os parâmetros que foram utilizados para cada uma das técnicas de Machine Learning na elaboração do modelo preditivo. É importante salientar que, na abordagem de Naive Bayes, foram testados três modelos diferentes, sendo selecionado, pelo sistema, o modelo que apresenta melhor desempenho, com base no F1 Score ponderado.

Tabela 7 - Parâmetros dos algoritmos de Machine Learning utilizados no modelo preditivo de classificação

Machine Learning		**Parâmetro**	**Intervalo de valores**	**Descrição**
Árvore de Decisão		criterion	[gini; entopy; log_loss]	Qualidade de uma divisão
		max_depth	[None; 3; 5; 10; 15; 20]	Profundidade máxima da árvore
		min_samples_split	[2; 5; 10; 15]	Número mínimo nó
		min_samples_leaf	[1; 2; 3; 4; 5; 6; 7; 10; 15]	Número mínimo folhas
Support Vector Machines		C	[0,1; 1; 10; 100; 1000]	Parâmetro de penalização do erro
		gama	[0,001; 0,01; 0,1; 1]	Coeficiente da função *kernel*
		kernel	[rbf; linear; poly; sigmoid]	Função
Naive Bayes	Gaussian	var_smoothing	10 números entre 10^{-10} e 10^{-1}	Suavização da variância
	Multimodal	alpha	[0,01; 0,1; 0,5; 1; 2; 5]	Suavização de Laplace
		fit_prior	[True; False]	Ajuste das probabilidades iniciais
	Bernoulli	alpha	[0,01; 0,1; 0,5; 1; 2; 5]	Suavização de Laplace
		binarize	[None; 0,5; 0; 1]	Limite para binarização
		fit_prior	[True; False]	Ajuste das probabilidades iniciais
Random Forest		n_estimators	[50; 100; 200; 300; 400]	Número de árvores

	max_depth	[None; 5; 10; 20; 30]	Profundidade máxima da árvore
	min_samples_split	[2; 5; 10; 15]	Número mínimo nó
	min_samples_leaf	[1; 2; 4; 6]	Número mínimo folhas
	max_features	[None; sqrt; log2]	Número de *features*
Regressão Logística	C	[0,001; 0,01; 0,1; 1; 10; 100; 1000; 10000]	Parâmetros de regularização
	penalty	[None; l1; l2: elasticnet]	Tipo de regularização
	solver	[newton-cg; lbfgs; liblinear; sag; saga]	Algoritmo de otimização
	max_iter	[100; 500; 1000; 2000]	Número máximo de interações
	class_weight	[None; balanced]	Peso das classes

As métricas de avaliação utilizadas para a comparação do desempenho dos algoritmos de Machine Learning foram a Accuracy, F1 Score ponderado, Precision, Recall e AUC (média One-vs-Rest). A Accuracy é um termo utilizado para descrever a precisão ou exatidão de um resultado, medida ou estimativa em relação a um valor de referência conhecido. Em termos simples, a Accuracy representa a proporção de previsões corretas feitas pela modelo em comparação com o total de previsões realizadas (Equação 12). O F1 Score ponderado calcula a média dos F1 Scores de cada classe. Por outras palavras, é uma métrica que tem em consideração a Precision e Recall das classes de forma equilibrada (Equações 13 e 14).

$$Accuracy = \frac{Número\ de\ previsões\ corretas}{Número\ total\ de\ previsões} \tag{12}$$

$$F1\ Score_{classe} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{13}$$

$$F1\ Score_{ponderado} = \frac{\sum_{i=1}^{n}\left(F1\ Score_{classe_i} \times Número\ de\ registos\ da\ classe_i\right)}{Número\ total\ de\ registos}\ , i \in classe \tag{14}$$

A Precision é uma métrica que se foca na exatidão das previsões positivas, mede a proporção entre os positivos identificados corretamente e todos os positivos do modelo (Equação 15). O Recall mede a capacidade do modelo identificar todos os exemplos positivos, ou seja, calcula a proporção de verdadeiros positivos em relação ao total de positivos reais (Equação 16). A

métrica AUC (Area Under the Curve) avalia o desempenho do modelo de classificação, tendo em conta a curva ROC (Receiver Operating Characteristic).

$$Precision = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ positivos} \quad (15)$$

$$Recall\ = \frac{Verdadeiros\ positivos}{Verdadeiros\ positivos + Falsos\ negativos} \quad (16)$$

Uma vez desenvolvido este modelo, foi aplicado o Ensemble Voting, uma técnica que combina os algoritmos de Machine Learning para efetuar uma previsão final. A implementação desta técnica teve o intuito de desenvolver um modelo mais eficaz e robusto. Porém, verificou-se que a sua aplicação não trouxe benefícios relevantes para o desempenho do modelo. Desta maneira, a descrição da sua implementação e os seus respetivos resultados obtidos, encontram-se apresentados no Anexo B.

Após a elaboração do modelo preditivo na aplicação Visual Studio Code e Jupyter Notebook, foi essencial o desenvolvimento do mesmo modelo na aplicação Orange[1], com o intuito de, numa fase posterior, elaborar uma comparação entre os desempenhos obtidos. O Orange utiliza um software de programação visual baseado em componentes para visualização de dados, Machine Learning, mineração de dados (*data mining*) e análise de dados. Nesta aplicação, a programação visual é implementada através de uma interface em que os fluxos de trabalho são criados ligando *widgets* predefinidos.

O modelo preditivo foi elaborado de modo a garantir uma comparação justa entre os dois modelos. Na aplicação Orange, o modelo é alimentado com os mesmos conjuntos de treino e teste obtidos na divisão estratificada realizada no modelo anterior, assegurando que ambos os modelos utilizem os mesmos dados. Além disso, foram utilizadas as mesmas técnicas de Machine Learning, sendo os parâmetros de cada algoritmo modificados de modo a coincidirem com os valores obtidos no modelo anterior. É relevante referir que alguns dos parâmetros considerados no outro modelo (Tabela 7) não estão disponíveis na aplicação Orange, pelo que apenas os parâmetros comuns foram configurados com os mesmos valores. As métricas de avaliação disponíveis na aplicação Orange são as mesmas que foram consideradas no modelo anterior. Na Figura 21, é apresentado o esquema visual da implementação do modelo preditivo de classificação na aplicação Orange.

[1] Disponível em: https://orangedatamining.com/

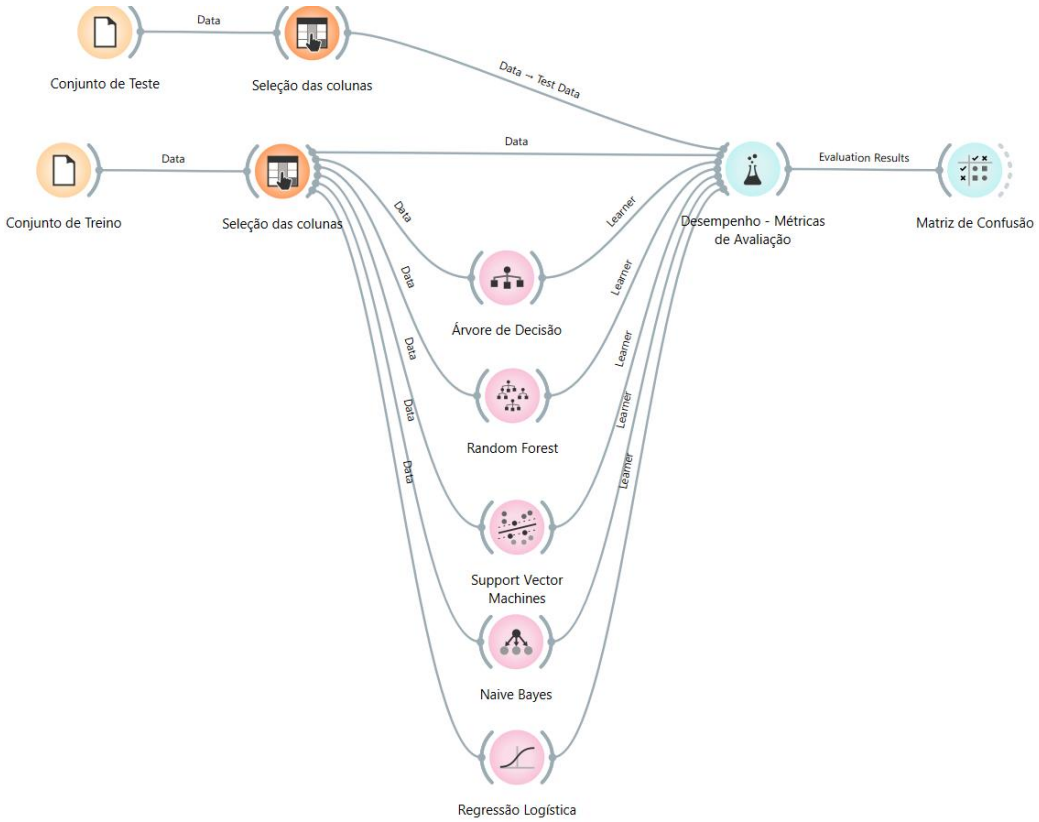


Figura 21 - Esquema visual do modelo preditivo na aplicação Orange

4.3 Avaliação

Os resultados das métricas de avaliação implementadas no modelo de classificação na aplicação Visual Studio Code são apresentados na Tabela 8.

Tabela 8 - Resultados das métricas de avaliação - Modelo preditivo de classificação - Aplicação Visual Studio Code

Machine Learning	Accuracy	F1 Score ponderado	Precision	Recall	AUC
Árvore de Decisão	0,6634	0,6665	**0,6949**	0,6634	**0,8847**
Support Vector Machines	0,5446	0,5432	0,5636	0,5446	0,5
Naive Bayes	0,3663	0,3382	0,3568	0,3663	0,6167
Random Forest	**0,67**	**0,6697**	0,6792	**0,67**	0,8803
Regressão Logística	0,3729	0,3733	0,4005	0,3729	0,577

Analisando a Tabela 8, verifica-se que o algoritmo Random Forest destaca-se, de forma geral, exibindo os melhores resultados em algumas métricas, nomeadamente, na Accuracy (0,67), no F1 Score ponderado (0,6697) e no Recall (0,67). No entanto, a técnica Árvore de Decisão também revela resultados bons, com valores muito próximos aos de Random Forest. Em comparação, esta técnica obteve os melhores de resultados nas métricas de Precision e AUC com valores de 0,6949 e 0,8847, respetivamente.

Em relação aos restantes algoritmos de Machine Learning, as técnicas Support Vector Machines, Naive Bayes e Regressão Logística apresentam desempenhos inferiores, em todas as métricas de avaliação. Destaca-se a metodologia Naive Bayes que, à exceção da métrica AUC, apresenta os valores mais baixos. A Regressão Logística é a técnica que exibe o menor número na métrica AUC.

Após esta análise, verifica-se que as técnicas de Árvore de Decisão e Random Forest apresentam os melhores resultados, demonstrando um desempenho superior em comparação com os restantes algoritmos. Posto isto, nas Figuras 22 e 23 são apresentadas as matrizes de confusão das técnicas Árvore de Decisão e Random Forest.

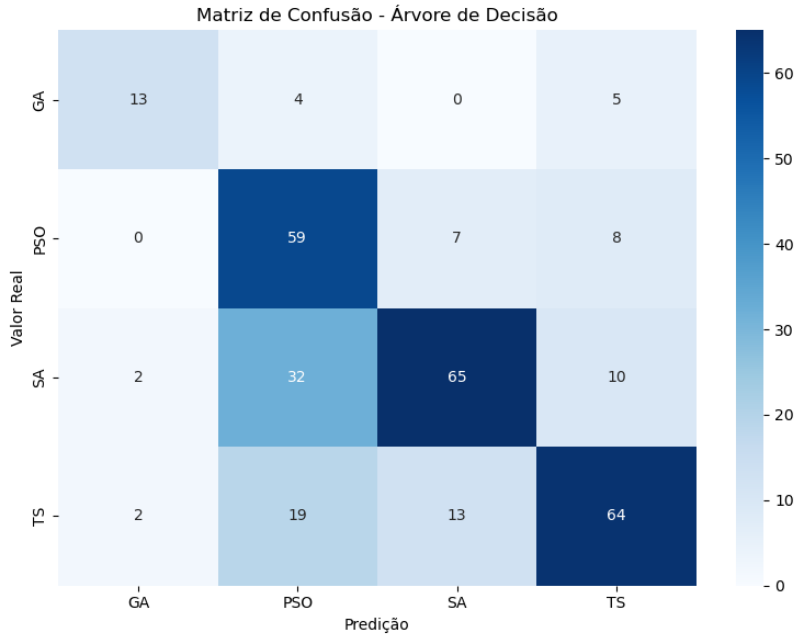


Figura 22 - Matriz de confusão técnica Árvores de Decisão - Visual Studio Code

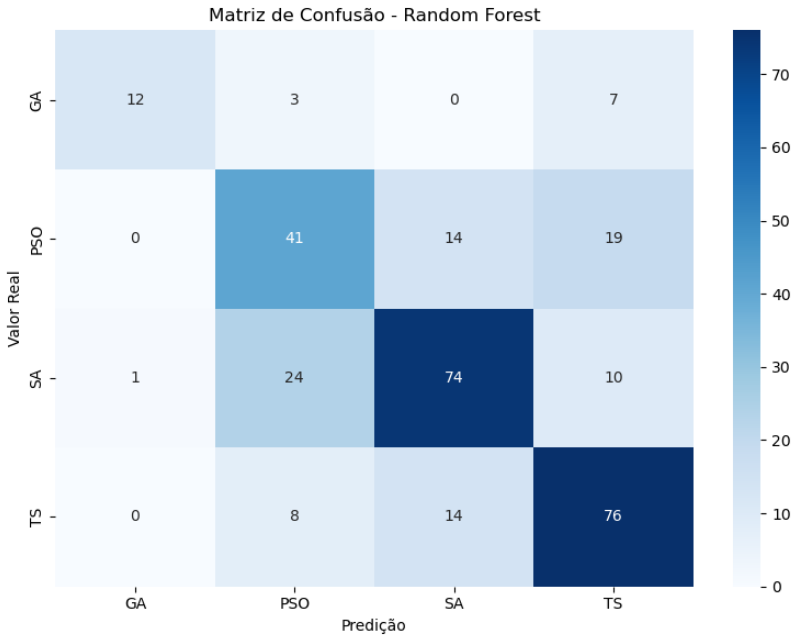


Figura 23 - Matriz de confusão técnica Random Forest - Visual Studio Code

As matrizes de confusão correspondem aos dados de teste. O conjunto de teste é composto por 303 registos e, como podemos observar pelas Figuras 22 e 23, a técnica Árvore de Decisão selecionou corretamente a Meta-heurística de 201 registos e o algoritmo Random Forest em 203 registos. A técnica Random Forest destaca-se na classe Tabu Search, ao alcançar uma taxa de classificação correta de 77,55%, correspondendo a 76 registos corretos num total de 98. É importante salientar que, ambos os algoritmos apresentam um bom desempenho geral, porém, ambos confundiram frequentemente a classe Simulated Annealing com o Particle Swarm Optimization.

Neste modelo, após a elaboração destas análises, verifica-se que a técnica Random Forest apresenta-se como o algoritmo que obteve o melhor desempenho global, exibindo resultados elevados nas métricas de avaliação implementadas. Contudo, a técnica Árvore de Decisão demonstra-se também como uma opção válida, embora apresente resultados ligeiramente inferiores ao do Random Forest.

Ao aplicar o mesmo modelo preditivo na plataforma Orange (Tabela 9), verifica-se que a técnica Árvore de Decisão obteve os melhores resultados em todas as métricas de avaliação, demonstrando um desempenho superior em relação aos restantes algoritmos. Esta metodologia alcançou o valor de 0,663 na Accuracy, 0,667 no F1 Score ponderado, 0,695 no Precision, 0,663 no Recall e 0,872 na AUC. O algoritmo Random Forest apresenta um desempenho consistente, exibindo valores próximos aos da Árvore de Decisão. Para as métricas de Accuracy, F1 Score ponderado, Precision, Recall e AUC obteve valores de 0,63, 0,63, 0,634, 0,63 e 0,856, respetivamente.

Em relação às técnicas Support Vector Machines, Naive Bayes e Regressão Logística, estas demonstram resultados de desempenho inferiores, destacando-se o algoritmo Support Vector

Machines que exibe os valores mais baixos em todas as métricas de avaliação, à exceção da métrica AUC. A Regressão Logística é o algoritmo que apresenta o valor mais baixo na métrica AUC.

Tabela 9 - Resultados das métricas de avaliação - Modelo preditivo de classificação - Aplicação Orange

Machine Learning	Accuracy	F1 Score ponderado	Precision	Recall	AUC
Árvore de Decisão	**0,663**	**0,667**	**0,695**	**0,663**	**0,872**
Support Vector Machines	0,33	0,325	0,383	0,33	0,667
Naive Bayes	0,459	0,466	0,488	0,459	0,693
Random Forest	0,63	0,63	0,634	0,63	0,856
Regressão Logística	0,409	0,343	0,331	0,409	0,616

Assim, tornou-se essencial a elaboração das matrizes de confusão das técnicas Árvore de Decisão e Random Forest, uma vez que ambas apresentam os melhores resultados, destacando-se dos restantes algoritmos. Nas Figuras 24 e 25 são apresentadas as matrizes de confusão das técnicas Árvore de Decisão e Random Forest elaboradas a partir dos dados do conjunto de teste.

	GA	PSO	SA	TS	∑
GA	13	4	0	5	**22**
PSO	0	59	7	8	**74**
SA	2	32	65	10	**109**
TS	2	19	13	64	**98**
∑	**17**	**114**	**85**	**87**	**303**

Figura 24 - Matriz de confusão técnica Árvores de Decisão - Aplicação Orange

	GA	PSO	SA	TS	∑
GA	13	3	0	6	**22**
PSO	0	37	16	21	**74**
SA	1	24	74	10	**109**
TS	1	8	22	67	**98**
∑	**15**	**72**	**112**	**104**	**303**

Figura 25 - Matriz de confusão técnica Random Forest - Aplicação Orange

Interpretando as matrizes de confusão, observa-se algumas diferenças entre o desempenho das técnicas Árvore de Decisão e Random Forest. O algoritmo Árvore de Decisão selecionou corretamente a Meta-heurística de 201 registos em 303, demonstrando uma boa capacidade de classificação, com destaque para a classe Particle Swarm Optimization, que apresenta 59 registos corretos em 74. No caso da técnica Random Forest, esta obteve um desempenho inferior em relação à Árvore de Decisão, apresentando um total de 191 registos selecionados corretamente. No entanto, esta técnica apresenta elevados números de classificações corretas nas classes Simulated Annealing e Tabu Search, com valores de 74 e 67, respetivamente. Em ambas as técnicas, é pertinente referir que, confundiram frequentemente a classe Simulated Annealing com Particle Swarm Optimization.

Considerando os resultados obtidos na aplicação Orange, conclui-se que a técnica Árvore de Decisão exibe resultados elevados nas métricas de avaliação, evidenciando-se como o algoritmo que obteve o melhor desempenho global. Porém, apesar de apresentar resultados ligeiramente inferiores, o algoritmo Random Forest revela-se também uma alternativa válida.

Uma vez interpretados os dados de desempenho do modelo preditivo na aplicação Visual Studio Code e Orange, verifica-se que a técnica Árvore de Decisão apresentou resultados iguais nas métricas de avaliação, bem como na matriz de confusão, demonstrando uma estabilidade nos seus resultados. Contudo, observam-se algumas diferenças no comportamento das restantes técnicas de Machine Learning. Os algoritmos Support Vector Machines e Random Forest, na plataforma Visual Studio Code, obtiveram um desempenho superior, enquanto as técnicas Naive Bayes e Regressão Logística apresentam melhores resultados na aplicação Orange.

Apesar das diferenças, tanto no Visual Studio Code como no Orange, as técnicas Árvore de Decisão e Random Forest destacam-se com os melhores resultados nos indicadores de desempenho. Porém, na aplicação Visual Studio Code, o algoritmo Random Forest obteve o melhor desempenho, enquanto na plataforma Orange, a técnica Árvore de Decisão apresentou-se mais eficaz. Considerando o desempenho geral de todas as técnicas, conclui-se que o modelo que demonstrou os melhores resultados foi o Random Forest na implementação realizada na aplicação Visual Studio Code.

4.4 Sumário

Neste capítulo são apresentadas todas as transformações efetuadas sobre os dados brutos, com o objetivo de construir uma base de dados adequada para aplicar no modelo preditivo de seleção de Meta-heurísticas. O processo de modelação é descrito de forma detalhada, incluindo a identificação das técnicas de Machine Learning utilizadas (Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest, Regressão Logística), bem como a explicação do processo de otimização dos respetivos hiperparâmetros. Adicionalmente, são especificadas as métricas de avaliação implementadas para avaliar o desempenho do modelo e é realizada uma análise dos resultados obtidos nos indicadores de desempenho.

5 Modelo preditivo – Afinação dos parâmetros das Meta-heurísticas

Este capítulo tem como objetivo expor as principais etapas envolvidas no desenvolvimento e avaliação do modelo preditivo elaborado para afinação dos parâmetros das Meta-heurísticas. O processo começa com a preparação dos dados, no qual os dados brutos são convertidos num formato adequado para a análise. Posteriormente, é apresentada a etapa da modelação, responsável pela definição das técnicas mais adequadas para o problema. Por último, efetua-se a avaliação dos resultados obtidos nas métricas de desempenho.

5.1 Preparação dos dados

Para a elaboração desta etapa foram utilizados como dados brutos a base de dados aplicada no desenvolvimento do modelo preditivo de seleção da Meta-heurística. Por outras palavras, recorreu-se à base de dados resultante das transformações descritas na secção 4.1. Esta escolha deve-se ao facto de que as transformações anteriores foram elaboradas com o intuito de selecionar apenas os registos mais relevantes e com maior qualidade. Além disso, considerando que este modelo está relacionado com o modelo preditivo de seleção da Meta-heurística, cujos temas são complementares, a utilização desta base de dados revelou-se essencial e apropriada.

Nesta etapa procedeu-se à filtragem dos dados por Meta-heurística, sendo aplicadas transformações diferentes a cada uma delas, de acordo com o número de parâmetros específicos e a sua dispersão de valores ao longo dos registos. Importa destacar que o intuito principal destas transformações foi assegurar a inclusão dos registos em que a Meta-heurística apresenta ocorrências com um valor igual ou o mais próximo possível do C_{max} ótimo (*makespan* ótimo) nas diferentes instâncias analisadas.

A Meta-heurística Particle Swarm Optimization é a técnica que contêm o maior número de parâmetros específicos (10), porém, apenas dois é que apresentam variações de valor ao longo

dos registos. Estes dois parâmetros são designados de 'NumPart' e 'NumItera' e correspondem ao número de partículas e iterações, respetivamente. Ambos apresentam uma ampla diversidade de valores ao longo dos registos. No entanto, especificamente no parâmetro 'NumPart', observou-se que certos valores surgem com uma maior frequência ao longo dos dados. Adicionalmente, verificou-se que muitos dos valores do parâmetro 'NumPart' associados aos registos em que a Particle Swarm Optimization obteve o melhor desempenho, em comparação com as restantes técnicas, coincidem com os valores mais recorrentes. Conforme o que foi observado, para a nova base de dados, foram selecionados os valores de 'NumPart' relativos às ocorrências da Particle Swarm Optimization que obtiveram um desempenho superior, relativamente às restantes técnicas (valores: 15, 23, 25, 26, 28, 33, 34, 35, 37, 71, 75, 77, 79). Porém, após esta transformação, identificou-se que quatro instâncias não estavam a ser representadas e foi fundamental incluir mais três valores de 'NumPart' (valores: 36, 76 e 100), correspondestes ao valor do parâmetro que permitiu a técnica alcançar o melhor C_{max} entre os registos disponíveis para essas instâncias. Na Tabela 10 é apresentado todos os valores do parâmetro 'NumPart' que foram incluídos na nova base de dados.

Tabela 10 - Valores do parâmetro 'NumPart' considerados na nova base de dados

Parâmetro 'NumPart'
15; 23; 25; 26; 28; 33; 34; 35; 36; 37; 71; 75; 76; 77; 79; 100

Uma vez concluídas as transformações (inclusão dos valores de 'NumPart' que contém o melhor desempenho na instância e dos valores de 'NumPart' das instâncias que não estavam representadas), constatou-se que a nova base de dados de Particle Swarm Optimization contém um número significativo de registos e uma grande diversidade de valores para o parâmetro 'NumItera'. Com o objetivo de reduzir o volume de dados, para cada instância, apenas metade dos registos presentes na base de dados original (secção 3.3) foi considerada na nova base de dados. Isto é, por exemplo, se uma instância apresentar 10 registos na base original, apenas os 5 com melhor desempenho, menor valor de C_{max}, serão incluídas na nova base de dados. Na Figura 26 é possível observar as transformações descritas.

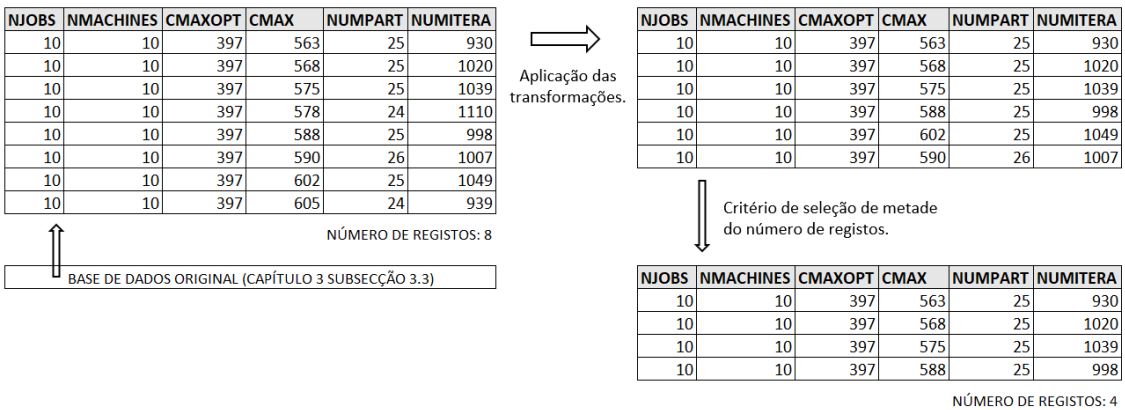


Figura 26 - Exemplo das transformações aplicadas a uma instância para a nova base de dados

A Meta-heurística Tabu Search possui quatro parâmetros, mas, contem apenas dois parâmetros que apresentam variações de valor ao longo dos registos. Estes parâmetros são designados de 'TabuListLen' e 'TS_StopCrit' e correspondem ao tamanho da lista Tabu e ao critério de paragem (número de iterações), respetivamente. O parâmetro 'TabuListLen' admite apenas os valores 1 e 3, enquanto 'TS_StopCrit' apresenta uma diversidade de valores possíveis. A realização de uma análise aos dados permitiu identificar a existência de algumas combinações de 'TabuListLen' e 'TS_StopCrit' que se repetem constantemente ao longo dos registos. Além disso, verificou-se que certas combinações recorrentes correspondem aos registos da técnica Tabu Search que apresentam o melhor desempenho em algumas instâncias, comparando com as restantes Meta-heurísticas. Desta forma, para cumprir o objetivo principal definido para as novas bases de dados, foram consideradas as combinações dos parâmetros correspondentes às instâncias em que a técnica Tabu Search obteve um desempenho superior. Contudo, constatou-se que uma instância não estava a ser representada na base de dados, sendo que, foi essencial a inclusão de uma nova combinação ('TabuListLen'=3; 'TS_StopCrit'=194). A combinação selecionada corresponde à configuração que apresentou melhor desempenho para a instância em questão, ou seja, a que obteve o menor valor de C_{max}. É importante referir que todos os registos associados a esta combinação foram adicionados na base de dados. As combinações dos parâmetros 'TabuListLen' e 'TS_StopCrit' considerados na construção desta base de dados são apresentadas na Tabela 11.

Tabela 11 - Combinações dos parâmetros 'TabuListLen' e 'TS_StopCrit' consideradas na nova base de dados

Parâmetro 'TabuListLen'	Parâmetro 'TS_StopCrit'
1	93; 97; 98; 99; 100; 101; 103; 104; 105; 106; 108; 109; 350
3	100; 194

A Meta-heurística Simulated Annealing é composta por quatro parâmetros, entre os quais três apresentam variações de valor ao longo dos registos. Os três parâmetros são denominados de 'InitTemp', 'NumIterak' e 'SA_StopCrit' e correspondem à temperatura inicial, número de iterações à mesma temperatura e citério de paragem (número de iterações), respetivamente. Como existe uma elevada variedade de valores nos parâmetros e não se verifica uma combinação predominante dos parâmetros entre os registos, foi essencial a implementação de uma abordagem seletiva que está alinhada com o objetivo definido para a construção das bases de dados. Posto isto, foram selecionadas as combinações dos parâmetros 'InitTemp', 'NumIterak' e 'SA_StopCrit' que estão associadas aos registos que demonstram os melhores resultados de C_{max} nas instâncias em que a Meta-heurística Simulated Annealing obteve o melhor desempenho, em comparação com as restantes técnicas. Na Tabela 12 são apresentadas as combinações que foram consideradas na base de dados.

Tabela 12 - Combinações dos parâmetros 'InitTemp', 'NumIterak' e 'SA_StopCrit' inseridas na nova base de dados

Parâmetro 'InitTemp'	Parâmetro 'NumIterak'	Parâmetro 'SA_StopCrit'
15	10	25
15	14	34
15	15	34; 35; 37; 38; 40; 44; 47; 50; 51; 52; 53
15	16	49; 50; 53; 55
15	17	38
15	30	200
15	33	165
14	13	33
14	15	38

No entanto, identificou-se três instâncias que não estavam representadas na base de dados. Posto isto, foram incluídas três combinações referidas na Tabela 13.

Tabela 13 - Combinações adicionais dos parâmetros 'InitTemp', 'NumIterk' e 'SA_StopCrit' inseridas na base de dados

Parâmetro 'InitTemp'	Parâmetro 'NumIterak'	Parâmetro 'SA_StopCrit'
15	25	100
15	29	188
15	31	190

A Meta-heurística Algoritmos Genéticos é composta por quatro parâmetros, dos quais apenas um, denominado 'NumGen' (número de gerações), apresenta variações de valor ao longo dos registos. Após uma análise, identificou-se que determinados valores de 'NumGen' aparecem com maior frequência, tendo sido incluídos na nova base de dados todos os registos associados aos três valores mais recorrentes: 100, 350 e 361. Além disso, foi essencial considerar os registos cujos valores de 'NumGen' estão relacionados com os melhores resultados de C_{max} nas instâncias em que a técnica Algoritmos Genéticos obteve o melhor desempenho, em comparação com as restantes Meta-heurísticas. Os valores integrados foram 106, 192, 200 e 349, sendo que os valores 100 e 350, já foram incluídos na etapa anterior. Contudo, após a aplicação destas transformações, verificou-se a ausência de uma instância na nova base de dados. De forma a garantir a sua representação, foram acrescentadas todas as ocorrências com o valor de 104 de 'NumGen'. Na Tabela 14 é apresentado todos os valores do parâmetro 'NumGen' que foram incluídos na nova base de dados.

Tabela 14 - Valores do parâmetro 'NumGen' considerados na nova base de dados

Parâmetro 'NumGen'
100; 104; 106; 192; 200; 349; 350; 361

Com a aplicação de todas as transformações, foram elaboradas quatro base de dados diferentes, cada uma associada a uma Meta-heurística. A base de dados relativa aos Algoritmos Genéticos possui 41 registos, a da Tabu Search contém 215, a da Particle Swarm Optimization é composta por 132 e a da Simulated Annealing apresenta 167 registos.

5.2 Modelação

Os modelos foram construídos utilizando a aplicação Visual Studio Code, com a implementação das ferramentas Python 3.11.4 e Jupyter Notebooks. Os modelos preditivos desenvolvidos são algoritmos de regressão, concebidos com o propósito de identificar os valores ideais dos parâmetros das Meta-heurísticas. Desta forma, foram aplicadas três técnicas de Machine Learning, nomeadamente: Árvores de Decisão, Random Forest e Regressão Linear.

5.2.1 Modelos preditivos individuais

Dado que a estruturação da base de dados foi realizada por Meta-heurística, procedeu-se ao desenvolvimento de modelos preditivos individuais específicos para cada técnica. Os modelos individuais foram elaborados utilizando a mesma estrutura metodológica, porém, diferenciam-se entre si pelo número e pelo tipo de variáveis dependentes consideradas. O modelo preditivo da Meta-heurística Algoritmos Genéticos possui uma única variável dependente, denominada 'NunGen'; o modelo da Tabu Search contém duas variáveis dependentes, 'TabuListLen' e 'TS_StopCrit'; o modelo da Particle Swarm Otimization inclui duas variáveis dependentes, 'NumPart' e 'NumItera'; o modelo Simulated Annealing apresenta três variáveis dependentes, 'InitTemp', 'NumIterak' e 'SA_StopCrit'.

Os modelos preditivos individuais utilizam as mesmas variáveis independentes: 'NJobs', 'NMachines', 'ObjFunc' e 'CmaxOpt'. As variáveis 'Cmax' e 'TimeExec' não foram incluídas nos modelos preditivos, uma vez que se trata de indicadores que só podem ser obtidos após a execução da Meta-heurística na instância correspondente com os parâmetros do registo. Os dados dos modelos individuais foram divididos em conjunto de treino e conjunto de teste numa proporção de 70/30, respetivamente. Na Figura 27 é possível observar o esquema geral utilizado na elaboração dos modelos preditivos individuais.

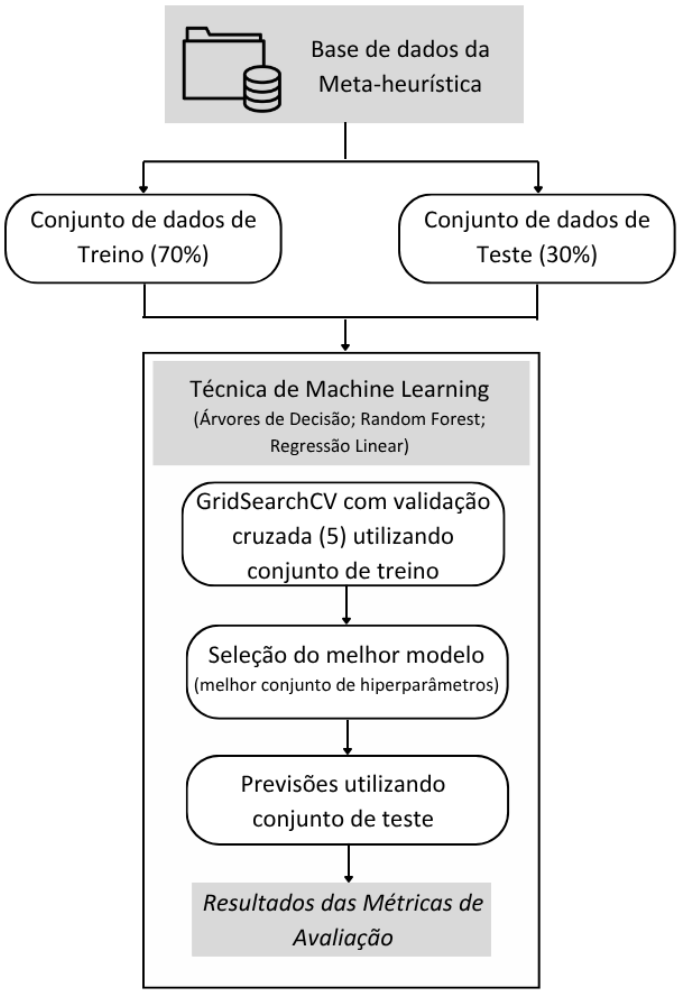


Figura 27 - Esquema geral dos modelos preditivos individuais de afinação de parâmetros das Meta-heurísticas

Na elaboração dos modelos preditivos, foi implementada a técnica GridSearchCV com o objetivo de identificar a combinação de hiperparâmetros que maximiza o desempenho das abordagens de Machine Learning. O processo GridSearchCV foi complementado com a aplicação da validação cruzada com um número de 5 *folds*, o que significa que, os dados são divididos em 5 partes, considerando que, em cada iteração, uma parte é utilizada para teste e as restantes para treino. A métrica de avaliação utilizada para selecionar o melhor conjunto de hiperparâmetros foi o valor negativo do Erro Quadrático Médio. A medida adotada fundamenta-se no facto de que o método GridSearchCV realiza a maximização da métrica, porém, para assegurar a qualidade do modelo, convém que o erro seja minimizado. Desta forma, ao aplicar o valor negativo do Erro Quadrático Médio, o GridSearchCV maximiza a métrica, resultando numa minimização do erro. Na Tabela 15 é possível observar os parâmetros que foram utilizados para cada uma das abordagens de Machine Learning na elaboração dos modelos preditivos individuais. É importante destacar que a Regressão Linear não possui hiperparâmetros, motivo pelo qual a técnica GridSearchCV não é implementada.

Tabela 15 - Parâmetros dos algoritmos de Machine Learning utilizados no modelo preditivo de regressão

Machine Learning	Parâmetro	Intervalo de valores	Descrição
Árvore de Decisão	criterion	[squared_error; absolute_error]	Qualidade de uma divisão
	max_depth	[None; 3; 5; 10; 15; 20]	Profundidade máxima da árvore
	min_samples_split	[2; 5; 10; 15]	Número mínimo nó
	min_samples_leaf	[1; 2; 3; 4; 5; 6; 7; 10; 15]	Número mínimo folhas
Random Forest	n_estimators	[50; 100; 200; 300; 400]	Número de árvores
	max_depth	[None; 5; 10; 20; 30]	Profundidade máxima da árvore
	min_samples_split	[2; 5; 10; 15]	Número mínimo nó
	min_samples_leaf	[1; 2; 4; 6]	Número mínimo folhas
	max_features	[None; sqrt; log2]	Número de *features*

As métricas de avaliação utilizadas para avaliar o desempenho dos modelos preditivos individuais foram o Erro Quadrático Médio (Mean Squared Error, MSE), Raiz do Erro Quadrático Médio (Root Mean Squared Error, RMSE), Erro Médio Absoluto (Mean Absolute Error, MAE), Erro Percentual Absoluto Médio (Mean Absolute Percentage Error, MAPE) e Coeficiente de Determinação (R-squared, R^2). O Erro Quadrático Médio é uma métrica estatística utilizada para medir a previsão dos modelos preditivos. Trata-se de uma medida que calcula a média dos quadrados das diferenças entre os valores previstos e os valores reais (Equação 17). A Raiz do Erro Quadrático Médio, como o nome indica, é a raiz quadrada do Erro Quadrático Médio (Equação 18).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(valor\ real_i - valor\ previsto_i)^2 \qquad ,n \in \mathbb{N} \qquad (17)$$

$$(RMSE) = \sqrt{Erro\ Quadrático\ Médio} \qquad (18)$$

O Erro Médio Absoluto mede a média da diferença absoluta entre os valores previstos pelo modelo e os valores reais (Equação 19). O Erro Percentual Absoluto Médio é uma métrica de avaliação expressa em percentagem e indica o quão distantes, em média, as previsões estão

dos valores reais (Equação 20). O Coeficiente de Determinação indica quanto da variabilidade dos valores reais é explicada pelo modelo preditivo (Equação 21).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|valor\ real_i - valor\ previsto_i| \quad , n \in \mathbb{N} \tag{19}$$

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{valor\ real_i - valor\ previsto_i}{valor\ real_i}\right| \quad , n \in \mathbb{N} \tag{20}$$

$$R^2 = 1 - \frac{\sum(valor\ real_i - valor\ previsto_i)^2}{\sum(valor\ real_i - média\ dos\ valores\ reais)^2} \tag{21}$$

Nos modelos individuais das Meta-heurísticas que contém mais do que uma variável dependente, as métricas de avaliação, à exceção da Raiz do Erro Quadrático Médio, são calculadas através da média dos valores alcançados por cada variável no respetivo modelo. É relevante referir que um modelo preditivo que apresenta um desempenho superior possui valores de Erro Quadrático Médio, Raiz do Erro Quadrático Médio, Erro Médio Absoluto e Erro Percentual Absoluto Médio reduzidos e o Coeficiente de Determinação é próximo de 1.

5.2.2 Modelo preditivo geral

Uma vez desenvolvidos os modelos preditivos individuais, foi elaborado um modelo preditivo geral que prevê os valores ideais dos parâmetros para as diferentes Meta-heurísticas. Este modelo utiliza uma base de dados que é composta pela combinação das quatro bases de dados individuais referentes a cada uma das Meta-heurísticas.

O modelo predito geral foi desenvolvido com o objetivo de prever, num único modelo, os parâmetros das quatro Meta-heurísticas consideradas neste estudo. Destaca-se pela sua capacidade de incorporar novas Meta-heurísticas de forma simples e eficiente, exigindo apenas pequenas alterações no modelo. Desta forma, não é necessário elaborar um modelo específico para cada nova Meta-heurística, como ocorre na abordagem dos modelos individuais.

A variável dependente do modelo geral é um vetor de três parâmetros, sendo que a composição varia consoante a Meta-heurística que está a ser aplicada. A elaboração do vetor de parâmetros exigiu a aplicação de uma abordagem padronizada, uma vez que as quatro Meta-heurísticas contêm diferentes conjuntos de parâmetros específicos. Assim, o vetor desenvolvido foi definido com uma estrutura de três posições ($[p1; p2; p3]$), onde cada posição retém um parâmetro relevante para a Meta-heurística em questão. Na Tabela 16 é possível observar a estrutura do vetor aplicada para cada Meta-heurística. Importa salientar que, no vetor, os parâmetros que não se aplicam à Meta-heurística são preenchidos com o valor de zero, ou seja, se a Meta-heurística em questão apenas apresentar um parâmetro específico, o vetor vai ter valor de zero em duas posições.

Tabela 16 - Estrutura utilizada na criação do vetor de parâmetros (variável dependente)

Meta-heurística	Estrutura
Algoritmos Genéticos	[‘NumGen’; 0; 0]
Tabu Search	[‘TS_StopCrit’; ‘TabuListLen’; 0]
Particle Swarm Optimization	[‘NumItera’; ‘NumPart’; 0]
Simulated Annealing	[‘Sa_StopCrit’; ‘NumIterak’; ‘InitTemp’]

As variáveis independentes implementadas no modelo geral incluem ‘MH’, ‘NJobs’, ‘NMachines’, ‘ObjFunc’ e ‘CmaxOpt’. Tal como nos modelos individuais, as variáveis ‘‘Cmax’ e ‘TimeExec’ não foram implementadas no modelo geral, dado que se trata de valores que só podem ser obtidos após a aplicação da Meta-heurística na instância correspondente. Os dados forma divididos de forma estratificada em conjunto de treino e conjunto de teste numa proporção de 70/30, respetivamente. A divisão estratificada foi realizada com base na variável ‘MH’, isto é, os dados do conjunto de treino e teste foram divididos garantindo que ambos os conjuntos contêm as mesmas proporções de dados para cada Meta-heurística. Na Figura 28 é possível observar o esquema geral utilizado na construção do modelo preditivo geral.

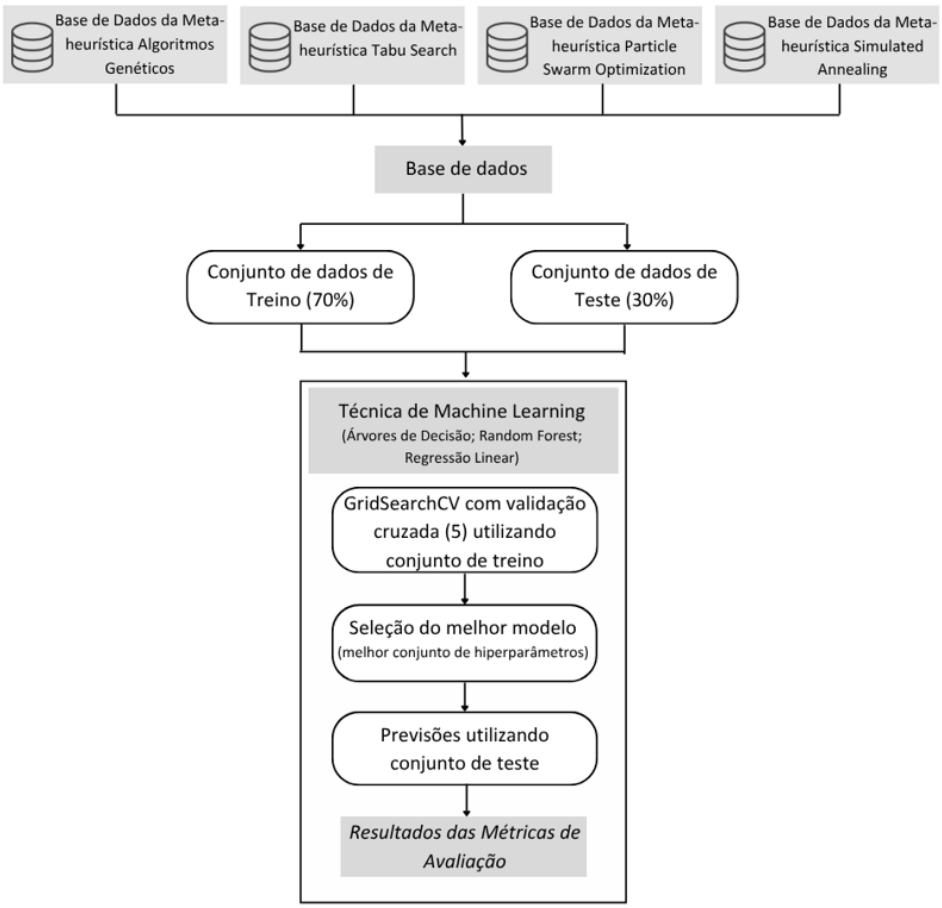


Figura 28 - Esquema geral do modelo preditivo geral de afinação de parâmetros das Meta-heurísticas

Tal como nos modelos preditivos individuais, no modelo preditivo geral foi implementado a técnica GridSearchCV para otimização dos hiperparâmetros das técnicas de Machine Learning. Além disso, o elemento de validação cruzada também foi aplicado com um número de 5 *folds* e a métrica de avaliação utilizada para selecionar o melhor conjunto de hiperparâmetros foi o valor negativo do Erro Quadrático Médio.

O modelo geral utiliza os algoritmos de Machine Learning combinados com a estratégia MultiOutputRegressor, permitindo a previsão simultânea dos parâmetros. Esta abordagem permite a elaboração de três modelos internos, um para cada parâmetro do vetor, uma vez que a variável dependente é um vetor composto por tês elementos. Isto significa que, ao utilizar, por exemplo, a técnica Árvore de Decisão, o sistema produzirá três árvores distintas, uma para cada componente do vetor.

O modelo preditivo geral é avaliado utilizando as mesmas métricas de avaliação implementadas nos modelos preditivos individuais. Como a variável dependente é composta por um vetor de três elementos, as métricas de avaliação são calculadas através da média dos valores de cada parâmetro, à exceção da Raiz do Erro Quadrático Médio.

5.3 Avaliação

Neste capítulo são apresentados os resultados das métricas de avaliação dos modelos preditivos individuais e do modelo preditivo geral, incluindo uma comparação entre as duas abordagens.

5.3.1 Avaliação dos modelos preditivos individuais

Os resultados das métricas de avaliação implementadas nos modelos preditivos individuais para as Meta-heurísticas Algoritmos Genéticos, Tabu Search, Particle Swarm Optimization e Simulated Annealing são apresentados nas Tabelas 17, 18, 19 e 20, respetivamente.

Tabela 17 - Resultados das métricas de avaliação - Modelo preditivo individual da Meta-heurística Algoritmos Genéticos

Machine Learning	MSE	RMSE	MAE	MAPE (%)	R^2
Árvore de Decisão	**14,3702**	**3,7908**	**2,7115**	**0,98**	**0,9991**
Random Forest	138,5621	11,7712	5,3146	3,96	0,9910
Regressão Linear	421,5533	20,5318	16,4200	14,22	0,9727

Tabela 18 - Resultados das métricas de avaliação - Modelo preditivo individual da Meta-heurística Tabu Search

Machine Learning	MSE	RMSE	MAE	MAPE (%)	R^2
Árvore de Decisão	73,1846	8,5548	2,0769	**2,21**	0,4237
Random Forest	**18,8335**	**4,3398**	**1,8461**	2,45	**0,9279**
Regressão Linear	266,3458	11,6436	9,0424	11,27	0,3934

Tabela 19 - Resultados das métricas de avaliação - Modelo preditivo individual da Meta-heurística Particle Swarm Optimization

Machine Learning	MSE	RMSE	MAE	MAPE (%)	R^2
Árvore de Decisão	31196,1483	176,6243	69,5098	8,16	0,6452
Random Forest	**29454,3671**	**171,6227**	**67,1109**	**7,75**	**0,6915**
Regressão Linear	41052,8317	202,6150	117,4866	13,96	0,5558

Tabela 20 - Resultados das métricas de avaliação - Modelo preditivo individual da Meta-heurística Simulated Annealing

Machine Learning	MSE	RMSE	MAE	MAPE (%)	R^2
Árvore de Decisão	**24,2258**	**4,9220**	**1,3767**	**3,04**	**0,5037**
Random Forest	33,5273	5,7903	1,5596	3,16	0,4914
Regressão Linear	39,6032	6,2931	3,4210	8,80	0,4589

O modelo preditivo individual correspondente à técnica Algoritmos Genéticos revela-se como o modelo mais robusto e com melhor desempenho, comparando com as restantes Meta-heurísticas. A implementação da técnica Árvore de Decisão apresenta os melhores resultados, indicando um valor de 14,3702 no Erro Quadrático Médio, 3,7908 na Raiz do Erro Quadrático Médio, 2,7115 no Erro Médio Absoluto e 0,98% no Erro Percentual Absoluto Médio. Para além disso, o Coeficiente de Determinação apresenta um valor de 0,9991, bastante próximo de 1, o que indica que o modelo conseguiu explicar quase toda a variabilidade presente na variável dependente.

Em contrapartida, o modelo preditivo desenvolvido para a Meta-heurística Particle Swarm Optimization demonstra o pior desempenho entre os quatro modelos. O algoritmo de Machine Learning que apresenta os melhores resultados é o Random Forest, porém, os valores das métricas de erro são elevados. A abordagem apresenta um valor de 29454,3671 no Erro Quadrático Médio, 171,6227 na Raiz do Erro Quadrático Médio, 67,7709 no Erro Médio Absoluto e 7,75% no Erro Percentual Absoluto Médio. Relativamente ao Coeficiente de

Determinação, o valor alcançado foi de 0,6915 que, apesar de não ser o mais baixo registado, seria preferível que estivesse próximo de 1.

O modelo individual da Meta-heurística Tabu Search exibe um desempenho eficiente, destacando-se no algoritmo Random Forest com 18,8335, 4,3398, 1,8461, 2,45% e 0,9279 nas métricas de avaliação Erro Quadrático Médio, Raiz do Erro Quadrático Médio, Erro Médio Absoluto, Erro Percentual Absoluto Médio e Coeficiente de Determinação. É importante referir que, para a métrica Erro Percentual Absoluto Médio, a abordagem de Árvore de Decisão obteve o melhor resultado de 2,21%.

O modelo individual relativo à Meta-heurística Simulated Annealing demonstra resultados intermédios. A técnica Árvore de Decisão apresenta o melhor desempenho, exibindo valores de 24,2258 no Erro Quadrático Médio, 4,9220 na Raiz do Erro Quadrático Médio, 1,3767 no Erro Médio Absoluto, 3,04% no Erro Percentual Absoluto Médio e 0,5037 no Coeficiente de Determinação. Apesar dos valores das métricas dos erros não serem elevados, o baixo valor do Coeficiente de Determinação (0,5037) evidencia que o modelo só consegue explicar metade da variabilidade dos dados da variável dependente.

A diferença entre os resultados do melhor e do pior modelo individual é bastante significativa. Conforme referido anteriormente, o modelo preditivo individual da Meta-heurística Algoritmos Genéticos com a abordagem Árvore de Decisão foi o que apresentou os melhores resultados nas métricas de avaliação. Porém, o modelo individual da Meta-heurística Particle Swarm Optimization com a técnica Random Forest foi o que obteve os piores resultados. É importante referir que, em ambos os casos, se trata das abordagens de Machine Learning que obtiveram o melhor desempenho para a respetiva Meta-heurística.

Esta diferença de resultados pode ser atribuída a dois fatores principais. Em primeiro lugar, o modelo individual que apresenta o melhor desempenho é composto apenas por uma variável dependente, o que reduz a complexidade do processo de modelação, sendo que, os restantes modelos preditivos individuais apresentam mais do que uma variável dependente. Em segundo lugar, esta discrepância pode estar relacionada com a dispersão e variabilidade dos dados das variáveis dependentes. Para tal, procedeu-se à avaliação dessa variação através de gráficos boxplot, do cálculo da amplitude interquartil e do desvio padrão. No caso do modelo individual da Meta-heurística Particle Swarm Optimization, que contém duas variáveis dependentes, apenas será analisada a variável 'NumItera', uma vez que, na análise das métricas de avaliação, esta obteve o pior desempenho, influenciando significativamente os resultados finais do modelo.

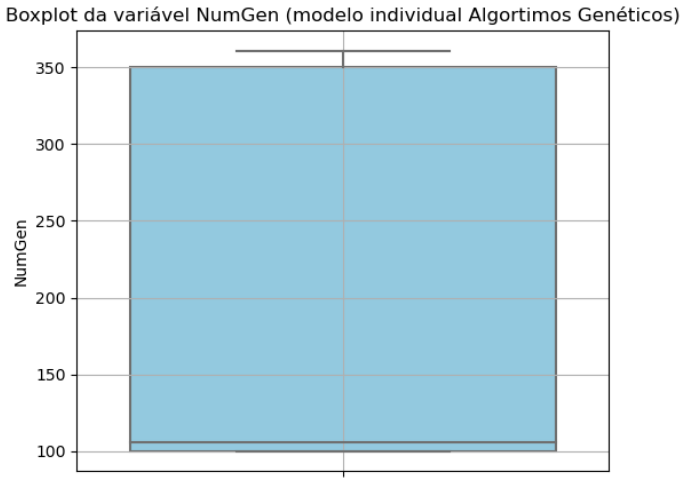


Figura 29 - Gráfico boxplot da variável 'NumGen'

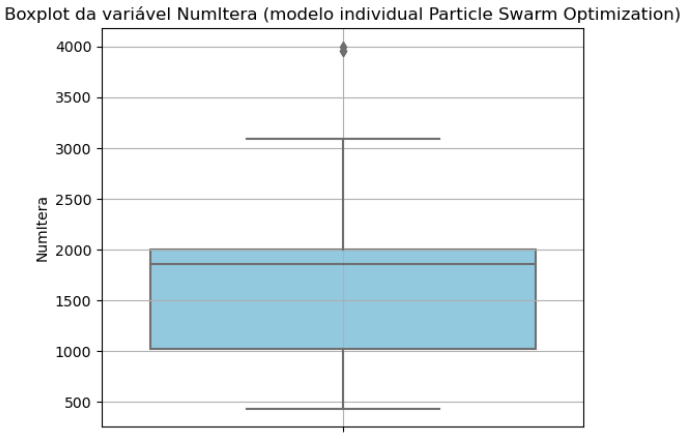


Figura 30 - Gráfico boxplot da variável 'NumItera'

Tabela 21 - Resultados da amplitude interquartil e do desvio padrão das variáveis 'NumGen' e 'NumItera'

Variável dependente	Amplitude interquartil	Desvio padrão
'NumGen' – Modelo individual Algoritmos Genéticos	250	119,82
'NumItera' – Modelo individual Particle Swarm Optimization	980	717,42

Analisando as Figuras 29 e 30, é possível observar, de forma clara, uma diferença significativa na dispersão e variabilidade dos dados entre as variáveis dependentes 'NumGen' e 'NumItera'. O boxplot correspondente à variável 'NumGen', que pertencente ao modelo preditivo individual da Meta-heurística Algoritmos Genéticos (Figura 29), demonstra uma distribuição mais concentrada dos dados, revelando uma baixa variabilidade. Este comportamento é

comprovado pelos valores da amplitude interquartil de 250 e do desvio padrão de 119,82, os quais são significativamente inferiores comparando com a outra variável.

O boxplot relativo à variável 'NumItera', correspondente ao modelo preditivo individual da Meta-heurística Particle Swarm Optimization (Figura 30), revela uma elevada dispersão dos dados, indicando uma grande variabilidade. Desta forma, apresenta valores de amplitude interquartil e de desvio padrão elevados, de 980 e 717,42, respetivamente.

Esta análise permite concluir que a alta variabilidade e dispersão da variável 'NumItera' dificulta o processo preditivo de modelo, contribuindo para um desempenho inferior, como é evidenciado nos resultados das métricas de avaliação. No entanto, a menor dispersão e variabilidade da variável dependente 'NumGen' permite que o modelo identifique padrões com maior facilidade, o que pode justificar o seu melhor desempenho nas métricas de avaliação.

Uma vez realizada a avaliação das métricas dos modelos preditivos individuais, tornou-se essencial obter a *feature importance* (importância das variáveis) dos modelos individuais com o melhor desempenho, com o objetivo de compreender melhor o seu funcionamento. Na Tabela 22 são apresentados os valores da *feature importance* dos modelos individuais das Meta-heurísticas Algoritmos Genéticos e Simulated Annealing, ambos com a técnica Árvore de Decisão e dos modelos individuais das Meta-heurísticas Tabu Search e Particle Swarm Optimization, desenvolvidos com a abordagem Random Forest.

Tabela 22 - *Feature importance* dos modelos individuais que apresentam o melhor desempenho

Modelo individual da Meta-heurística	Variável 'NJobs'	Variável 'NMachines'	Variável 'CmaxOpt'	Variável 'ObjFunc'
Algoritmos Genéticos	0,9998	0	0,0002	0
Tabu Search	0,3118	0,0136	0,6123	0,0623
Particle Swarm Optimization	0,5975	0,0003	0,3928	0,0094
Simulated Annealing	0,8492	0,0012	0,1496	0

A *feature importance* é uma técnica que avalia e classifica as variáveis de um conjunto de dados com base na influência que exercem na variável(is) dependente(s). Isto é, quantifica o quanto cada variável contribui para a realização das decisões do modelo. Quanto maior for o valor da *feature importance* mais relevância esta variável tem na previsão da variável dependente. Analisando a Tabela 22, verifica-se que a variável 'NJobs' assume um papel importante na maioria dos modelos individuais. No modelo preditivo individual da Meta-heurística Algoritmos Genéticos esta variável apresenta uma importância quase absoluta, com valor de 0,9998. Porém, de forma semelhante, no modelo correspondente à técnica Simulated Annealing, a variável 'NJobs' apresenta um valor elevado de 0,8492.

No modelo preditivo individual correspondente à Meta-heurística Particle Swarm Optimization é possível observar uma distribuição mais equilibrada ao longo das variáveis independentes apresentadas. A variável 'NJobs' continua a exercer uma importância superior (0,5975), com a variável 'CmaxOpt' em segundo lugar (0,3928). Em contrapartida, o modelo individual referente à Meta-heurística Tabu Search apresenta a variável 'CmaxOpt' como sendo a mais relevante do modelo (0,6123), seguida pela variável 'NJobs' (0,3118).

Os quatro modelos preditivos individuais não apresentam valores elevados de importância na variável 'NMachines'. Na realidade, os valores exibidos são bastante pequenos, próximos de 0. No entanto, é possível verificar que, em todos os modelos preditivos individuais, a variável 'Njobs' contém sempre uma importância relativa. Esta análise permite entender quais são os fatores mais determinantes que influenciam a configuração dos parâmetros de cada Meta-heurística.

5.3.2 Avaliação do modelo preditivo geral

Os resultados das métricas de avaliação implementadas no modelo preditivo geral são apresentados na Tabela 23.

Tabela 23 - Resultados das métricas de avaliação - Modelo preditivo geral

Machine Learning	MSE	RMSE	MAE	MAPE (%)	R^2
Árvore de Decisão	3557,0479	59,6410	**11,7545**	**2,75**	0,9762
Random Forest	**3515,9281**	**59,2953**	12,0319	3,09	**0,9766**
Regressão Linear	134857,3693	367,2293	142,7425	250,03	0,4582

Verifica-se que o modelo preditivo geral baseado na abordagem Random Forest apresenta menor valor de Erro Quadrático Médio, com 3515,9281, indicando também menor valor de Raiz do Erro Quadrático Médio, com valor de 59,2953. Para além disto, esta técnica exibe o melhor desempenho na métrica Coeficiente de Determinação, com um valor de 0,9766. Relativamente à métrica Erro Médio Absoluto e Erro Percentual Absoluto Médio, a técnica Árvores de Decisão destaca-se, apresentando os melhores resultados, com valores de 11,7545 e 2,75%, respetivamente.

O modelo preditivo geral baseado na abordagem Regressão Linear demonstra o pior desempenho geral, com valores significativamente superiores em todas as métricas de avaliação. Estes resultados sugerem que esta técnica não é a mais adequada para o tipo de dados utilizados no modelo preditivo geral. Por outro lado, o modelo construído com a técnica Random Forest destaca-se como o mais eficaz, demonstrando um desempenho geral superior nas métricas de avaliação. Para aprofundar o funcionamento deste modelo, na Tabela 24 são apresentados os valores de *feature importance* das variáveis independentes.

Tabela 24 - Feature importance do modelo preditivo geral baseado em Random Forest

Variável 'MH'	Variável 'NJobs'	Variável 'NMachines'	Variável 'CmaxOpt'	Variável 'ObjFunc'
0,8585	0,0972	0,0001	0,0427	0,0015

No modelo preditivo geral, a variável 'MH' apresenta uma importância relativa no modelo, com um valor de *feature importance* de 0,8585. Esta variável revela-se como a principal responsável pela capacidade preditiva do modelo. A variável 'NJobs' surge em segundo lugar, com um valor de 0,0972, substancialmente inferior ao da variável 'MH'. As variáveis 'CmaxOpt' e 'ObjFunc' apresentam valores bastante baixos de importância, de 0,0427 e 0,0015, respetivamente. A variável 'NMachines', tal como observado nos modelos preditivos individuais, demonstra um valor quase nulo, próximo de 0.

5.3.3 Comparação

Após a análise dos modelos preditivos individuais e do modelo preditivo geral, procedeu-se à elaboração de uma comparação entre eles. Para assegurar uma comparação equilibrada, calculou-se a média dos valores das métricas de avaliação dos modelos individuais, considerando apenas os modelos individuais que obtiveram o melhor desempenho em cada Meta-heurística. Isto é, foram considerados os modelos preditivos individuais das Meta-heurísticas Algoritmos Genéticos e Simulated Annealing, ambos com a abordagem Árvore de Decisão e os modelos das Meta-heurísticas Tabu Search e Particle Swarm Optimization, desenvolvidos com a técnica Random Forest. A comparação foi realizada apenas com o modelo preditivo geral que apresentou o melhor desempenho, o que utiliza a abordagem Random Forest.

Tabela 25 - Resultados das métricas de avaliação - Comparação entre o modelo preditivo geral e os modelos preditivos individuais

Modelo	MSE	RMSE	MAE	MAPE (%)	R^2
Modelo Preditivo Geral (com abordagem Random Forest)	**3515,9281**	**59,2953**	**12,0319**	**3,09**	**0,9766**
Média dos Modelos Preditivos Individuais	7377,9415	85,8950	18,2613	3,56	0,7806

Ao comparar o desempenho do modelo preditivo individual com o modelo preditivo geral (Tabela 25), verifica-se que o modelo geral apresenta resultados significativamente melhores

em todas as métricas de avaliação. O valor do Erro Quadrático Médio do modelo geral (3515,9281) é significativamente mais baixo do que o dos modelos individuais (7377,9415), atingindo cerca de metade do valor dos modelos individuais. Consequentemente, o valor da métrica Raiz do Erro Quadrático Médio do modelo geral (59,2953) é inferior ao valor dos modelos individuais (85,8950).

Relativamente às métricas Erro Médio Absoluto e Erro Percentual Absoluto Médio, o modelo geral apresenta um desempenho superior, demonstrando valores inferiores, de 12,0319 e 3,09%, respetivamente. É importante salientar que o valor do Erro Percentual Médio Absoluto referente aos modelos individuais é a métrica que menos diferença contém do valor do modelo preditivo geral, apresentando um valor de 3,56%.

A métrica Coeficiente de Determinação apresenta um valor superior no modelo preditivo geral, com um valor de 0,9766, bastante próximo de 1, indicando que o modelo consegue de forma eficaz descrever o comportamento da variável dependente com base nas variáveis independentes.

Com base na análise comparativa, conclui-se que o modelo preditivo geral apresenta um desempenho superior, revelando-se como o modelo mais preciso e eficiente, proporcionando previsões mais robustas. Estimava-se que os modelos individuais, por apenas considerarem uma única Meta-heurística, apresentassem melhor resultados, principalmente no modelo relativo à Meta-heurística Particle Swarm Optimization. Este modelo apresentou valores de erro significativamente elevados, o que pode ser justificado pela complexidade do modelo e, sobretudo, pela elevada dispersão e variabilidade dos valores das variáveis dependentes. Estes valores afetam diretamente o cálculo de média das métricas de avaliação, contribuindo para um desempenho inferior dos modelos preditivos individuais quando comparados com o modelo preditivo geral.

O modelo preditivo geral, ao utilizar um conjunto mais abrangente de dados, consegue identificar padrões com maior eficácia, apresentando resultados com melhor desempenho. Para além disso, o modelo preditivo contém uma maior adaptabilidade, proporcionando a possibilidade de incorporar novas Meta-heurísticas de forma simples, através da realização de pequenas adaptações no modelo.

5.4 Sumário

Neste capítulo é abordado a etapa de preparação de dados, a qual inclui a aplicação de transformações aos dados brutos com o objetivo de construir uma base de dados adequada para a fase da modelação. Na etapa de modelação são descritas todas as técnicas utilizadas na elaboração dos modelos preditivos individuais e no modelo preditivo geral. Na elaboração dos modelos foram aplicadas três técnicas de Machine Learning (Árvores de Decisão, Random Forest, Regressão Linear) e, em todas as abordagens, é realizada a otimização dos

hiperparâmetros. Por último, este capítulo apresenta uma avaliação do desempenho dos modelos preditivos com base nas métricas de avaliação identificadas.

6 Conclusão

O escalonamento de produção é uma atividade crítica no âmbito operacional das empresas, influenciando a sua produtividade e qualidade. Trata-se de um problema de natureza dinâmica e alta complexidade, cujo objetivo é elaborar um plano que atinja as metas de produção, respeitando todas as restrições operacionais e técnicas. As Meta-heurísticas podem ser utilizadas para a resolução de problemas de escalonamento, apresentando como vantagem a capacidade de fornecerem soluções satisfatórias em tempos de execução razoáveis.

As Meta-heurísticas, apesar de serem abordagens de fácil implementação, apresentam dificuldades. Particularmente, a escolha da técnica mais adequada para a resolução de um determinado problema é uma tarefa complexa, visto que é difícil eleger uma Meta-heurística como a melhor de todas. A eficácia de cada Meta-heurística depende das propriedades específicas do problema em questão. Além disto, as Meta-heurísticas possui parâmetros que necessitam de ser definidos, cuja configuração influência o desempenho e qualidade das soluções.

As técnicas de Machine Learning podem ser incorporadas às Meta-heurísticas com o intuito de aprimorar o desempenho dos sistemas de produção. Esta metodologia contribui para a seleção da Meta-heurística mais adequada para a resolução de um problema, bem como para a afinação eficiente dos seus parâmetros.

Assim, o principal objetivo deste trabalho consistiu em oferecer uma contribuição significativa para a resolução de problemas de escalonamento de produção. Para tal, foi elaborado um sistema de seleção e autoparametrização de Meta-heurísticas, que utiliza técnicas de Machine Learning para otimizar o escalonamento de tarefas. O modelo desenvolvido é aplicado exclusivamente a problemas do tipo Job-Shop e a parametrização das Meta-heurísticas é realizado de forma *offline*. O sistema utiliza técnicas de aprendizagem supervisionada, incluindo Árvores de Decisão, Support Vector Machines, Naives Bayes, Random Forest, Regressão Linear e Regressão Logística.

6.1 Principais conclusões

O sistema de seleção e autoparametrização de Meta-heurísticas com recurso a técnicas de Machine Learning foi estruturado em dois modelos preditivos distintos. O primeiro modelo foi desenvolvido para identificar a Meta-heurística mais adequada para um problema de escalonamento, enquanto o segundo modelo é responsável pela afinação dos parâmetros da Meta-heurística selecionada.

O modelo preditivo relativo ao processo de seleção da Meta-heurística foi desenvolvido com a aplicação Visual Studio Code, com recurso ao Python 3.11.4 e Jupyter Notebook. Dado que se trata de um algoritmo de classificação, foram consideradas cinco técnicas de Machine Learning: Árvores de Decisão, Support Vector Machines, Naive Bayes, Random Forest e Regressão Logística. Para otimizar os hiperparâmetros de cada abordagem de Machine Learning e maximizar o desempenho do modelo, foi implementada a técnica GridSearchCV, em conjunto com a validação cruzada de 10 *folds*.

O modelo preditivo de classificação (seleção da Meta-heurística) foi igualmente desenvolvido na aplicação Orange, uma ferramenta *low-code* baseada em programação visual, com o intuito de comparar os desempenhos obtidos. Para assegurar uma comparação justa entre os dois ambientes, os modelos foram alimentados com os mesmos conjuntos de treino e teste e foram utilizadas as mesmas técnicas de Machine Learning. Além disso, os hiperparâmetros das abordagens de Machine Learning foram configurados com os mesmos valores em ambas as implementações.

A partir da análise dos resultados das métricas de avaliação, verificou-se que no modelo preditivo desenvolvido na aplicação Visual Studio Code, a abordagem Random Forest destaca-se exibindo o melhor desempenho. Por outro lado, no modelo preditivo elaborado na plataforma Orange, a técnica Árvores de Decisão foi a que demonstrou um desempenho mais eficaz. Apesar de as diferenças nos valores das métricas de avaliação entre os dois ambientes não sejam muito significativas, o modelo com Random Forest implementado no Visual Studio Code destacou-se por apresentar um desempenho geral superior.

O modelo preditivo destinado à afinação de parâmetros das Meta-heurísticas, classificado como um algoritmo de regressão, foi desenvolvido a partir de duas abordagens distintas. Ambos os modelos foram construídos na plataforma Visual Studio Code, com a aplicação das ferramentas Python 3.11.4 e Jupyter Notebooks. O desenvolvimento dos modelos baseou-se unicamente em três técnicas de Machine Learning: Árvores de Decisão, Random Forest e Regressão Linear. Para maximizar o desempenho e otimizar os hiperparâmetros destas técnicas, recorreu-se à aplicação do GridSearchCV, combinado com a validação cruzada de 5 *folds*.

A primeira abordagem adotada baseou-se na criação de modelos preditivos individuais para cada uma das Meta-heurísticas consideradas no estudo: Algoritmos Genéticos, Tabu Search, Particle Swarm Optimization e Simulated Annealing. Embora todos os modelos sigam a mesma estrutura metodológica, distinguem-se pelo número e tipo de variáveis dependentes utilizadas.

Cada modelo individual é alimentado com os dados correspondentes à respetiva Meta-heurística e é capaz de prever apenas os parâmetros dessa técnica específica.

A segunda abordagem, por sua vez, teve como objetivo a criação de um modelo preditivo geral, capaz de prever, num único modelo, os parâmetros das quatro Meta-heurísticas. Este modelo apresenta uma vantagem significativa, uma vez que a inclusão de novas Meta-heurísticas pode ser realizada de forma simples, exigindo apenas a elaboração de pequenas mudanças no modelo existente, sem ter a necessidade de criar um novo modelo individual, como é solicitado na abordagem anterior. Nesta abordagem foi necessário aplicar um processo padronizado que permitisse a definição da variável dependente. Esta variável consiste num vetor de três elementos, preenchidos de forma distinta por cada Meta-heurística. Devido ao tipo de variável dependente considerada no modelo preditivo geral, foi essencial a combinação da estratégia MultiOutputRegressor com os algoritmos de Machine Learning.

Na avaliação dos modelos preditivos individuais (primeira abordagem), cada modelo foi avaliado com base na técnica de Machine Learning utilizada. Verificou-se que os melhores desempenhos foram obtidos pelos modelos associados às Meta-heurísticas Algoritmos Genéticos e Simulated Annealing, ambos com a técnica Árvores de Decisão e pelos modelos relativos às Meta-heurísticas Tabu Search e Particle Swarm Optimization, desenvolvidos com o algoritmo Random Forest. Comparando os resultados de cada modelo individual, o modelo da Meta-heurística Algoritmos Genéticos destacou-se como o mais eficaz, evidenciando valores baixos de erro e um R^2 (Coeficiente de Determinação) bastante elevado, próximo de 1. Por outro lado, o modelo correspondente à Meta-heurística Particle Swarm Optimization registou os piores resultados, com valores de erro significativamente muitos altos em todas as métricas de avaliação.

Esta diferença de desempenho pode ser atribuída à complexidade dos modelos e à variabilidade e dispersão dos valores das variáveis dependentes. O modelo da Meta-heurística Algoritmos Genéticos contém apenas uma variável dependente, o que contribui para uma menor complexidade. Já o modelo de Particle Swarm Optimization possui duas variáveis dependentes, sendo que uma delas, o número de iterações, apresenta uma elevada dispersão e variabilidade, o que influência a capacidade preditiva do modelo. A análise da amplitude interquartil demonstra que a variável número de iterações (do modelo de Particle Swarm Optimization) apresenta um valor de 74,45% superior ao da variável número de gerações (do modelo de Algoritmos Genéticos). Além disso, o desvio padrão da variável número de gerações representa apenas 16,70% do desvio padrão do número de iterações. Estes fatores ajudam a entender o desempenho inferior obtido pelo modelo de Particle Swarm Optimization em relação ao modelo individual de Algoritmos Genéticos.

Na análise do modelo preditivo geral (segunda abordagem), constatou-se que a versão baseada na técnica Random Forest obteve o melhor desempenho global, destacando-se com a mais eficaz. Com a conclusão das avaliações em ambas as abordagens, procedeu-se à realização de uma comparação entre os dois tipos de modelos desenvolvidos, ou seja, entre os modelos preditivos individuais e o modelo preditivo geral. Para assegurar uma comparação equilibrada,

foi calculada a média dos resultados das métricas de avaliação dos modelos preditivos individuais (primeira abordagem), considerando apenas os modelos que apresentaram o melhor desempenho para cada Meta-heurística. Da mesma forma, nesta comparação, utilizou-se exclusivamente o modelo preditivo geral (segunda abordagem) baseado no Random Forest, uma vez que apresentava melhores indicadores de desempenho.

Os resultados da comparação revelaram que o modelo preditivo geral com Random Forest supera, em termos de desempenho geral, os modelos preditivos individuais. Especificamente, obteve-se uma redução significativa no Erro Quadrático Médio (MSE), com uma melhoria de 52,35% e um aumento de 20,07% no Coeficiente de Determinação (R^2), em relação aos modelos preditivos individuais.

A partir dos resultados alcançados e das análises realizadas ao longo do desenvolvimento do sistema de seleção e autoparametrização das Meta-heurísticas, é possível responder às questões de investigação inicialmente propostas (secção 1.3). Relativamente à primeira questão *"De que modo os algoritmos de Machine Learning contribuem para prever os melhores parâmetros para diferentes instâncias de problemas de escalonamento?"*, os resultados demonstram que a integração de Machine Learning no sistema desempenha um papel fundamental, tanto na seleção da Meta-heurística mais adequada como na definição dos parâmetros ideias a serem aplicados. O modelo preditivo consegue, através da análise das caraterísticas do problema (como o número de tarefas e máquinas) identificar a Meta-heurística que apresenta uma maior probabilidade de obter um bom desempenho. Após a seleção da Meta-heurística, os algoritmos de Machine Learning são novamente utilizados para determinar os seus parâmetros. Desta forma, a incorporação das abordagens de Machine Learning permite obter uma configuração mais automática e eficiente, reduzindo significativamente a intervenção manual e o tempo computacional necessário em abordagens tradicionais de parametrização.

Quanto à segunda questão *"Quais são os principais desafios e limitações na implementação de um sistema de seleção e autoparametrização de Meta-heurísticas em problemas de escalonamento?"*, verificam-se algumas limitações relevantes, sobretudo na fase de preparação de dados, uma vez que o desempenho dos modelos preditivos é influenciado pela qualidade dos dados. Particularmente, os dados utilizados no processo de treino têm de ser representativos, de modo a permitir que o modelo realize previsões com eficácia. A ausência da representatividade pode provocar *overfitting*, isto é, o modelo adapta-se demasiado aos dados de treino, comprometendo a sua capacidade preditiva. Além disso, a variabilidade e dispersão dos valores das variáveis nas instâncias de escalonamento representam um desafio, uma vez que este comportamento compromete a identificação de padrões, afetando negativamente o desempenho do modelo.

6.2 Limitações e trabalho futuro

Na realização deste trabalho foram detetadas algumas limitações subjacentes ao desenvolvimento do sistema de seleção e autoparametrização das Meta-heurísticas que utiliza Machine Learning para otimizar o escalonamento de tarefas. Uma das limitações enfrentadas foi a dificuldade em obter uma base de dados adequada que permitisse obter bons resultados de desempenho no sistema. Embora já existisse uma base de dados inicial, foi necessário realizar um trabalho rigoroso de preparação de dados, que envolveu a filtragem dos registos de forma estratégica, de modo a garantir a qualidade necessária para que os modelos pudessem ser construídos. Ambas as bases de dados utilizadas neste trabalho (uma destinada à seleção da Meta-heurística e outra à parametrização das Meta-heurísticas) foram desenvolvidas com o objetivo de apenas incluir os registos que apresentavam melhor qualidade. Para isso, foram elaborados diversos subsets (subconjuntos de dados), sendo selecionadas aquelas que exibiam uma maior eficácia. Este processo de preparação de dados revelou-se bastante importante, uma vez que uma base de dados mal estruturada influência negativamente a capacidade preditiva do modelo.

Outra limitação identificada foi a dispersão e variabilidade dos dados da base de dados utilizada. Conforme mencionado anteriormente, o modelo individual desenvolvido para a parametrização da Meta-heurística Particle Swarm Optimization teve um desempenho inferior em relação aos outros modelos individuais preditivos. Isso ocorreu devido à elevada dispersão e variabilidade dos valores de uma das variáveis dependentes considerada no modelo. Esta variável foi comparada com outra variável dependente utilizada em outro modelo e verificou-se uma diferença significativa nos valores de amplitude interquartil e de desvio padrão, sendo que a variável dependente do modelo Particle Swarm Optimization apresentava os valores maiores. Durante este projeto, não foi possível reduzir esta dispersão de a forma melhorar o desempenho do modelo. No entanto, esta questão seria um desafio que poderia ser resolvido num trabalho futuro. Este trabalho futuro teria com propósito a diminuição da dispersão e variabilidade dos valores da variável dependente, mantendo, ao mesmo tempo, uma quantidade suficiente de variabilidade para que o modelo consiga aprender padrões relevantes para realizar as previsões.

Um outro possível trabalho futuro, que não foi elaborado nesta dissertação por questões de tempo, consiste na criação de um sistema capaz de selecionar e identificar os parâmetros das Meta-heurísticas quando o utilizador insere uma nova instância de escalonamento que não esteja presente na base de dados. Para tal, o sistema teria de analisar a nova instância de escalonamento, identificar a instância mais semelhante nos registos existentes e, com base nesta semelhança, apresentar a Meta-heurística mais adequada e os seus respetivos parâmetros.

Outro trabalho futuro que poderia ser elaborado seria, a partir do sistema desenvolvido nesta dissertação, criar um modelo preditivo capaz de estimar os valores variáveis 'Cmax' e 'TimeExec', com base na Meta-heurística selecionada e nos seus respetivos parâmetros. A variável 'Cmax' representa o valor de *makespan* obtido através da aplicação da Meta-heurística

e dos seus parâmetros, enquanto a variável ‘TimeExec’ corresponde ao tempo de execução necessário para obter a solução.

Bibliografia

Agrawal, P. *et al.* (2021) 'Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019)', *IEEE Access*, 9. Available at: https://doi.org/10.1109/ACCESS.2021.3056407.

Alhijawi, B. and Awajan, A. (2023) 'Genetic algorithms: theory, genetic operators, solutions, and applications', *Evolutionary Intelligence*, 17, pp. 1245–1256. Available at: https://doi.org/10.1007/S12065-023-00822-6/FIGURES/9.

Alorf, A. (2023) 'A survey of recently developed metaheuristics and their comparative analysis', *Engineering Applications of Artificial Intelligence*, 117. Available at: https://doi.org/10.1016/J.ENGAPPAI.2022.105622.

Boussaïd, I., Lepagnot, J. and Siarry, P. (2013) 'A survey on optimization metaheuristics', *Information Sciences*, 237, pp. 82–117. Available at: https://doi.org/10.1016/J.INS.2013.02.041.

Calvet, L. *et al.* (2017) 'Learnheuristics: Hybridizing metaheuristics with machine learning for optimization with dynamic inputs', *Open Mathematics*. Available at: https://doi.org/10.1515/math-2017-0029.

Carrilho, L.M., Oliveira, F. and Hamacher, S. (2024) 'A novel exact formulation for parallel machine scheduling problems', *Computers & Chemical Engineering*, 184, p. 108649. Available at: https://doi.org/10.1016/J.COMPCHEMENG.2024.108649.

Cervantes, J. *et al.* (2020) 'A comprehensive survey on support vector machine classification: Applications, challenges and trends', *Neurocomputing*, 408, pp. 189–215. Available at: https://doi.org/10.1016/J.NEUCOM.2019.10.118.

Chauhan, N.K. and Singh, K. (2019) 'A Review on Conventional Machine Learning vs Deep Learning', in *2018 International Conference on Computing, Power and Communication Technologies, GUCON 2018*, pp. 347–352. Available at: https://doi.org/10.1109/GUCON.2018.8675097.

Chen, R. *et al.* (2020) 'A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem', *Computers & Industrial Engineering*, 149. Available at: https://doi.org/10.1016/J.CIE.2020.106778.

Coelho, P. and Silva, C. (2021) 'Parallel Metaheuristics for Shop Scheduling: enabling Industry 4.0', *Procedia Computer Science*, 180, pp. 778–786. Available at: https://doi.org/10.1016/J.PROCS.2021.01.328.

Couronné, R., Probst, P. and Boulesteix, A.L. (2018) 'Random forest versus logistic regression: A large-scale benchmark experiment', *BMC Bioinformatics*, 19. Available at: https://doi.org/10.1186/s12859-018-2264-5.

Cuevas, E., Zaldívar, D. and Pérez, M. (2022) 'Metaheuristic schemes and machine learning techniques: A synergistic perspective', *Applied Mathematical Modelling*, 104, pp. 850–851. Available at: https://doi.org/10.1016/J.APM.2021.12.025.

Desale, S. *et al.* (2015) 'Heuristic and Meta-Heuristic Algorithms and Their Relevance to the Real World: A Survey', *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS*, 2(5).

Destouet, C. *et al.* (2023) 'Flexible job shop scheduling problem under Industry 5.0: A survey on human reintegration, environmental consideration and resilience improvement', *Journal of Manufacturing Systems*, 67, pp. 155–173. Available at: https://doi.org/10.1016/J.JMSY.2023.01.004.

Dokeroglu, T., Deniz, A. and Kiziloz, H.E. (2022) 'A comprehensive survey on recent metaheuristics for feature selection', *Neurocomputing*, 494, pp. 269–296. Available at: https://doi.org/10.1016/J.NEUCOM.2022.04.083.

van Engelen, J.E. and Hoos, H.H. (2019) 'A survey on semi-supervised learning', *Machine Learning*, 109, pp. 373–440. Available at: https://doi.org/10.1007/s10994-019-05855-6.

Fausto, F. *et al.* (2019) 'From ants to whales: metaheuristics for all tastes', *Artificial Intelligence Review* , 53, pp. 753–810. Available at: https://doi.org/10.1007/S10462-018-09676-2.

Fawagreh, K., Gaber, M.M. and Elyan, E. (2014) 'Random forests: From early developments to recent advancements', *Systems Science and Control Engineering*, 2, pp. 602–609. Available at: https://doi.org/10.1080/21642583.2014.956265.

Fernanda, N. and Utamima, A. (2024) 'Solving Preemptive Single Machine Scheduling with Lovebird Algorithm and Constructive Heuristics for Equal-Length Jobs', *Procedia Computer Science*, 234, pp. 310–317. Available at: https://doi.org/10.1016/J.PROCS.2024.03.005.

Fernandez-Viagas, V., Talens, C. and Prata, B. de A. (2024) 'A speed-up procedure and new heuristics for the classical job shop scheduling problem: A computational evaluation', *European Journal of Operational Research*, 322(3), pp. 783–794. Available at: https://doi.org/10.1016/J.EJOR.2024.11.026.

Fuchigami, H.Y. and Rangel, S. (2018) 'A survey of case studies in production scheduling: Analysis and perspectives', *Journal of Computational Science*, 25, pp. 425–436. Available at: https://doi.org/10.1016/j.jocs.2017.06.004.

Gallo, C. and Capozzi, V. (2019) 'A Simulated Annealing Algorithm for Scheduling Problems', *Journal of Applied Mathematics and Physics*, 07(11). Available at: https://doi.org/10.4236/jamp.2019.711176.

Genuer, R. *et al.* (2017) 'Random Forests for Big Data', *Big Data Research*, 9, pp. 28–46. Available at: https://doi.org/10.1016/J.BDR.2017.07.003.

Georgiadis, G.P., Elekidis, A.P. and Georgiadis, M.C. (2019) 'Optimization-based scheduling for the process industries: From theory to real-life industrial applications', *Processes*, 7(7). Available at: https://doi.org/10.3390/pr7070438.

Glover, F. (1986) 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research*, 13(5), pp. 533–549. Available at: https://doi.org/10.1016/0305-0548(86)90048-1.

Gogna, A. and Tayal, A. (2013) 'Metaheuristics: Review and application', *Journal of Experimental and Theoretical Artificial Intelligence*, 25(4), pp. 503–526. Available at: https://doi.org/10.1080/0952813X.2013.782347.

Hajji, M.K., Hamlaoui, O. and Hadda, H. (2024) 'A simulated annealing metaheuristic approach to hybrid flow shop scheduling problem', *Advances in Industrial and Manufacturing Engineering*, 9. Available at: https://doi.org/10.1016/J.AIME.2024.100144.

Hiran, K.K. et al. (2021) Machine Learning: Master Supervised and Unsupervised Learning Algorithms with Real Examples (English Edition). BPB Publications.

Holland, J. (1975) *Adaptation in natural and artificial systems*. Ann Arbor, Michigan, United States: University of Michigan Press.

Huang, C., Li, Y. and Yao, X. (2020) 'A Survey of Automatic Parameter Tuning Methods for Metaheuristics', *IEEE Transactions on Evolutionary Computation*. Available at: https://doi.org/10.1109/TEVC.2019.2921598.

Huynh, T.N., Do, D.T.T. and Lee, J. (2021) 'Q-Learning-based parameter control in differential evolution for structural optimization', *Applied Soft Computing*, 107. Available at: https://doi.org/10.1016/J.ASOC.2021.107464.

Jadhav, S.D. and Channe, H.P. (2016) 'Comparative Study of K-NN, Naive Bayes and Decision Tree Classification Techniques', *International Journal of Science and Research (IJSR)*, 5(1). Available at: https://doi.org/10.21275/v5i1.nov153131.

Janiesch, C., Zschech, P. and Heinrich, K. (2021) 'Machine learning and deep learning', *Electronic Markets*, 31, pp. 685–695. Available at: https://doi.org/10.1007/s12525-021-00475-2.

Kar, A.K. (2016) ‘Bio inspired computing – A review of algorithms and scope of applications’, *Expert Systems with Applications*, 59, pp. 20–32. Available at: https://doi.org/10.1016/J.ESWA.2016.04.018.

Karaboga, D. (2005) ‘An idea based on Honey Bee Swarm for Numerical Optimization’, *Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department* [Preprint].

Karimi-Mamaghan, M. *et al.* (2022) ‘Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art’, *European Journal of Operational Research*, 296(2), pp. 393–422. Available at: https://doi.org/10.1016/J.EJOR.2021.04.032.

Katoch, S., Chauhan, S.S. and Kumar, V. (2020) ‘A review on genetic algorithm: past, present, and future’, *Multimedia Tools and Applications*, 80, pp. 8091–8126. Available at: https://doi.org/10.1007/s11042-020-10139-6.

Kavitha, S., Varuna, S. and Ramya, R. (2017) ‘A comparative analysis on linear regression and support vector regression’, in *Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016*. IEEE. Available at: https://doi.org/10.1109/GET.2016.7916627.

Kennedy, J. and Eberhart, R. (1995) ‘Particle swarm optimization’, *Proceedings of ICNN’95 - International Conference on Neural Networks* [Preprint]. Available at: https://doi.org/10.1109/ICNN.1995.488968.

Kiran, M.S. and Findik, O. (2015) ‘A directed artificial bee colony algorithm’, *Applied Soft Computing*, 26, pp. 454–462. Available at: https://doi.org/10.1016/J.ASOC.2014.10.020.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) ‘Optimization by simulated annealing’, *Science*, 220(4598), pp. 671–680. Available at: https://doi.org/10.1126/science.220.4598.671.

Klein, N., Gnägi, M. and Trautmann, N. (2024) ‘Mixed-integer linear programming for project scheduling under various resource constraints’, *European Journal of Operational Research*, 319(1), pp. 79–88. Available at: https://doi.org/10.1016/J.EJOR.2024.06.036.

Komaki, G.M., Sheikh, S. and Malakooti, B. (2018) ‘Flow shop scheduling problems with assembly operations: a review and new trends’, *International Journal of Production Research*, 57(10), pp. 2926–2955. Available at: https://doi.org/10.1080/00207543.2018.1550269.

Lei, D. and Cai, J. (2020) ‘Multi-population meta-heuristics for production scheduling: A survey’, *Swarm and Evolutionary Computation*, 58. Available at: https://doi.org/10.1016/J.SWEVO.2020.100739.

Li, X. and Yang, G. (2016) ‘Artificial bee colony algorithm with memory’, *Applied Soft Computing*, 41, pp. 362–372. Available at: https://doi.org/10.1016/J.ASOC.2015.12.046.

Mahesh, B. (2020) 'Machine Learning Algorithms - A Review', *International Journal of Science and Research (IJSR)*, 9(1), p. 381. Available at: https://doi.org/10.21275/ART20203995.

Maier, H.R. *et al.* (2019) 'Introductory overview: Optimization using evolutionary algorithms and other metaheuristics', *Environmental Modelling & Software*, 114, pp. 195–213. Available at: https://doi.org/10.1016/J.ENVSOFT.2018.11.018.

Montero, E., Riff, M.C. and Neveu, B. (2014) 'A beginner's guide to tuning methods', *Applied Soft Computing*, 17, pp. 39–51. Available at: https://doi.org/10.1016/J.ASOC.2013.12.017.

Mraihi, T., Driss, O.B. and EL-Haouzi, H.B. (2024) 'Distributed Permutation Flow Shop Scheduling Problem with Worker flexibility: Review, trends and model proposition', *Expert Systems with Applications*, 238. Available at: https://doi.org/10.1016/J.ESWA.2023.121947.

Müller, D. *et al.* (2022) 'An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning', *European Journal of Operational Research*, 302(3), pp. 874–891. Available at: https://doi.org/10.1016/J.EJOR.2022.01.034.

Niroumandrad, N., Lahrichi, N. and Lodi, A. (2024) 'Learning tabu search algorithms: A scheduling application', *Computers & Operations Research*, 170. Available at: https://doi.org/10.1016/J.COR.2024.106751.

Pereira, I. *et al.* (2021) 'A hybrid metaheuristics parameter tuning approach for scheduling through racing and case-based reasoning', *Applied Sciences (Switzerland)*, 11. Available at: https://doi.org/10.3390/app11083325.

Pereira, I.A. (2014) *Sistema Inteligente para Escalonamento Assistido por Aprendizagem*. Universidade de Trás-os-Montes e Alto Douro.

Pikalov, M. and Pismerov, A. (2023) 'Exploratory Landscape Analysis Based Parameter Control', *GECCO 2023 Companion - Proceedings of the 2023 Genetic and Evolutionary Computation Conference Companion*, pp. 2378–2381. Available at: https://doi.org/10.1145/3583133.3596364.

Piotrowski, A.P., Napiorkowski, J.J. and Piotrowska, A.E. (2020) 'Population size in Particle Swarm Optimization', *Swarm and Evolutionary Computation*, 58. Available at: https://doi.org/10.1016/J.SWEVO.2020.100718.

Radhika, S. and Chaparala, A. (2018) 'Optimization using evolutionary metaheuristic techniques: a brief review', *Brazilian Journal of Operations & Production Management*, 15(1). Available at: https://doi.org/10.14488/bjopm.2018.v15.n1.a17.

Ray, S. (2019) 'A Quick Review of Machine Learning Algorithms', in Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends,

Prespectives and Prospects, COMITCon 2019. IEEE. Available at: https://doi.org/10.1109/COMITCon.2019.8862451.

Reijnen, R. *et al.* (2023) 'Learning to Adapt Genetic Algorithms for Multi-Objective Shop Scheduling Problems', *GECCO 2023 Companion - Proceedings of the 2023 Genetic and Evolutionary Computation Conference Companion*, pp. 315–318. Available at: https://doi.org/10.1145/3583133.3590700/SUPPL_FILE/P315-REIJNEN-SUPPL.ZIP.

Rossit, D.A., Tohmé, F. and Frutos, M. (2018) 'The Non-Permutation Flow-Shop scheduling problem: A literature review', *Omega*, 77, pp. 143–153. Available at: https://doi.org/10.1016/J.OMEGA.2017.05.010.

Ruiz, R., Pan, Q.K. and Naderi, B. (2019) 'Iterated Greedy methods for the distributed permutation flowshop scheduling problem', *Omega*, 83, pp. 213–222. Available at: https://doi.org/10.1016/J.OMEGA.2018.03.004.

Sadeghian, Z. *et al.* (2023) 'A review of feature selection methods based on meta-heuristic algorithms', *Journal of Experimental & Theoretical Artificial Intelligence*, pp. 1–51. Available at: https://doi.org/10.1080/0952813X.2023.2183267.

Santos, A.S., Madureira, A.M. and Varela, L.R. (2022) 'A Self-Parametrization Framework for Meta-Heuristics', *Mathematics*, 10(3). Available at: https://doi.org/10.3390/math10030475.

Saravanan, R. and Sujatha, P. (2018) 'A State of Art Techniques on Machine Learning Algorithms: A Perspective of Supervised Learning Approaches in Data Classification', *International Conference on Intelligent Computing and Control Systems, ICICCS 2018*, pp. 945–949. Available at: https://doi.org/10.1109/ICCONS.2018.8663155.

Sarker, I.H. (2021) 'Machine Learning: Algorithms, Real-World Applications and Research Directions', *SN Computer Science*. Available at: https://doi.org/10.1007/s42979-021-00592-x.

Schröer, C., Kruse, F. and Gómez, J.M. (2021) 'A Systematic Literature Review on Applying CRISP-DM Process Model', *Procedia Computer Science*, 181, pp. 526–534. Available at: https://doi.org/10.1016/J.PROCS.2021.01.199.

Scornet, E., Biau, G. and Vert, J.P. (2015) 'Consistency of random forests', *The Annals of Statistics*, 43(4). Available at: https://doi.org/10.1214/15-AOS1321.

Shami, T.M. *et al.* (2022) 'Particle Swarm Optimization: A Comprehensive Survey', *IEEE Access*, 10. Available at: https://doi.org/10.1109/ACCESS.2022.3142859.

Singh Kushwah, J. *et al.* (2022) 'Comparative study of regressor and classifier with decision tree using modern tools', *Materials Today: Proceedings*, 56(6), pp. 3571–3576. Available at: https://doi.org/10.1016/J.MATPR.2021.11.635.

Sivananda, M. and Kumar, G.K. (2024) 'Classification and Regression Based on Decision Tree Algorithm for Machine Learning', *International Journal of Scientific Research in Engineering and Management*, 8(2).

Strassl, S. and Musliu, N. (2022) 'Instance space analysis and algorithm selection for the job shop scheduling problem', *Computers & Operations Research*, 141. Available at: https://doi.org/10.1016/J.COR.2021.105661.

Taheri, S. and Mammadov, M. (2013) 'Learning the naive bayes classifier with optimization models', *International Journal of Applied Mathematics and Computer Science*, 23(4). Available at: https://doi.org/10.2478/amcs-2013-0059.

Talbi, E.-G. (2021) 'Machine Learning into Metaheuristics: A Survey and Taxonomy', *ACM Computing Surveys*. Available at: https://doi.org/10.1145/3459664.

Tatsis, V.A. and Parsopoulos, K.E. (2020) 'Reinforced Online Parameter Adaptation Method for Population-based Metaheuristics', in *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*. IEEE. Available at: https://doi.org/10.1109/SSCI47803.2020.9308488.

Telikani, A. *et al.* (2021) 'Evolutionary Machine Learning: A Survey', *ACM Computing Surveys*, pp. 1–35. Available at: https://doi.org/10.1145/3467477.

Thomas Rincy, N. and Gupta, R. (2020) 'A Survey on Machine Learning Approaches and Its Techniques':, *2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2020* [Preprint]. Available at: https://doi.org/10.1109/SCEECS48394.2020.190.

Tondut, J. *et al.* (2022) 'An automatic kriging machine learning method to calibrate meta-heuristic algorithms for solving optimization problems', *Engineering Applications of Artificial Intelligence*, 113. Available at: https://doi.org/10.1016/J.ENGAPPAI.2022.104940.

Verbraeken, J. *et al.* (2020) 'A Survey on Distributed Machine Learning', *ACM Computing Surveys*. Available at: https://doi.org/10.1145/3377454.

Wang, L., Pan, Z. and Wang, J. (2021) 'A Review of Reinforcement Learning Based Intelligent Optimization for Manufacturing Scheduling', *Complex System Modeling and Simulation*, 1(4), pp. 257–270. Available at: https://doi.org/10.23919/CSMS.2021.0027.

Xiong, H. *et al.* (2022) 'A survey of job shop scheduling problem: The types and models', *Computers & Operations Research*, 142. Available at: https://doi.org/10.1016/J.COR.2022.105731.

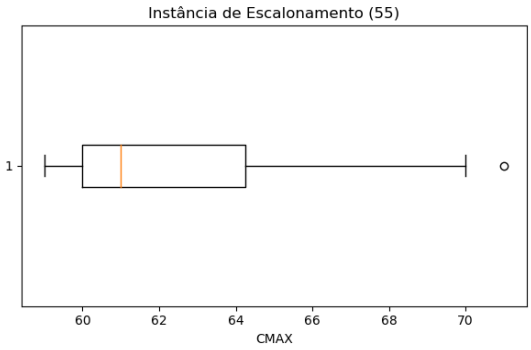
Xu, W. *et al.* (2022) 'Optimization approaches for solving production scheduling problem: A brief overview and a case study for hybrid flow shop using genetic algorithms', *Advances in Production Engineering And Management*, pp. 45–56. Available at: https://doi.org/10.14743/apem2022.1.420.

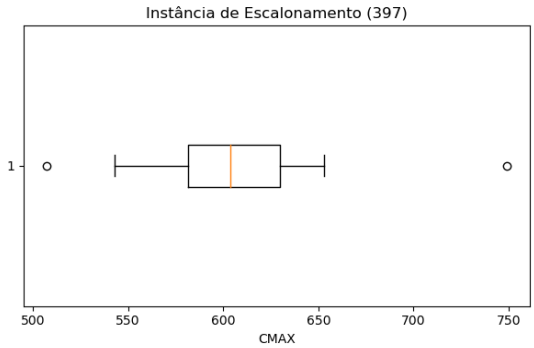
Yang, Z. *et al.* (2025) 'A Q-learning-based improved multi-objective genetic algorithm for solving distributed heterogeneous assembly flexible job shop scheduling problems with transfers', *Journal of Manufacturing Systems*, 79, pp. 398–418. Available at: https://doi.org/10.1016/J.JMSY.2025.02.002.

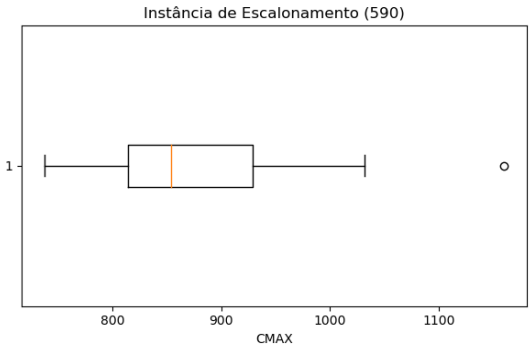
Zhang, X. and Zhu, G.Y. (2024) 'A literature review of reinforcement learning methods applied to job-shop scheduling problems', *Computers & Operations Research*, 175. Available at: https://doi.org/10.1016/J.COR.2024.106929.

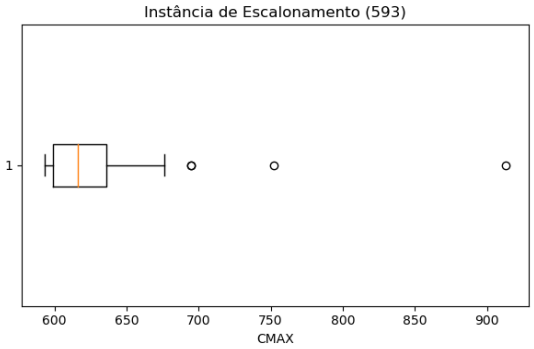
Anexo A

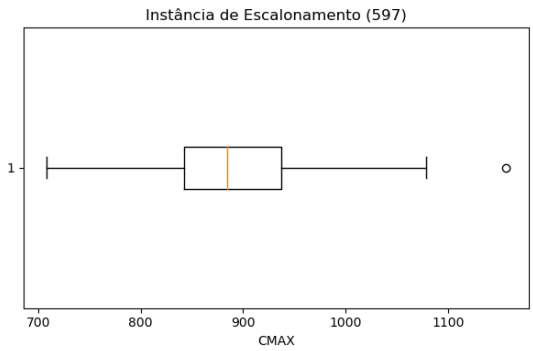
Apresentação dos gráficos boxplot das instâncias de escalonamento que contêm *outliers*.

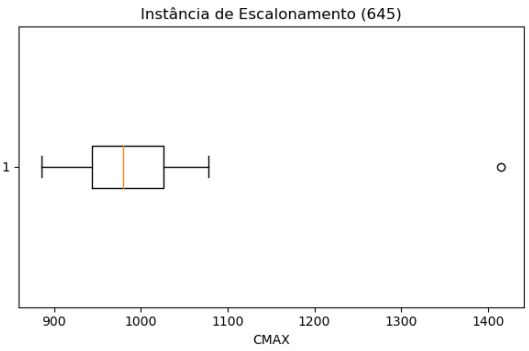


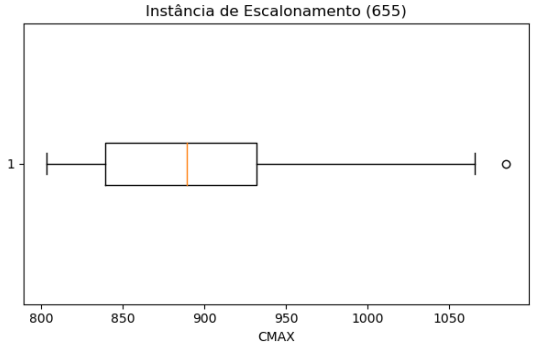


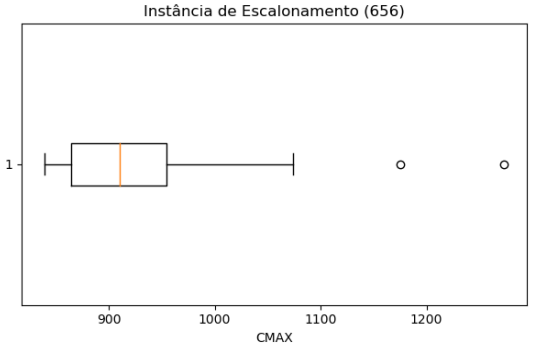


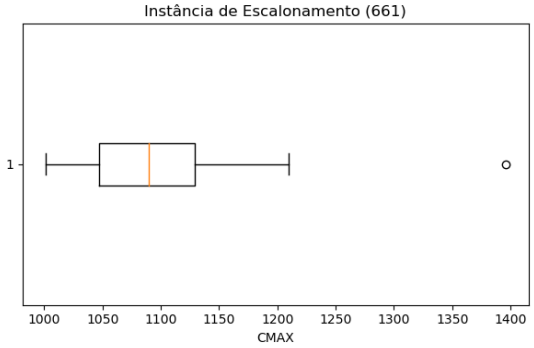


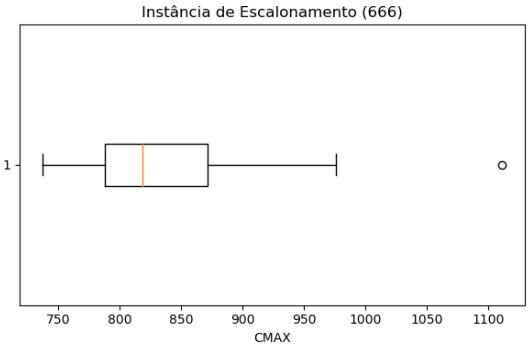


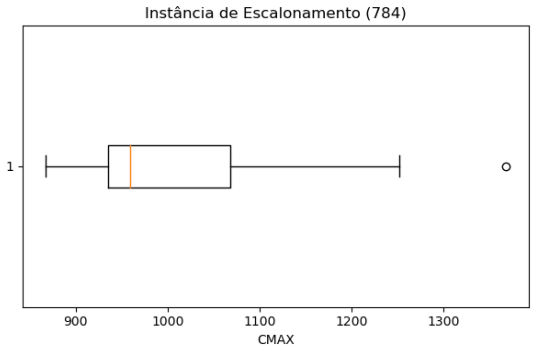


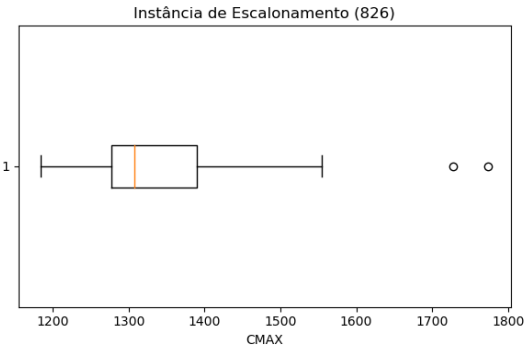


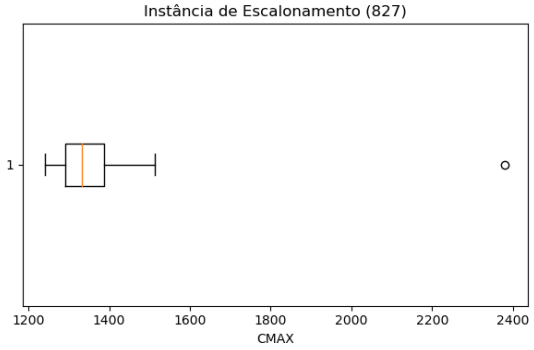


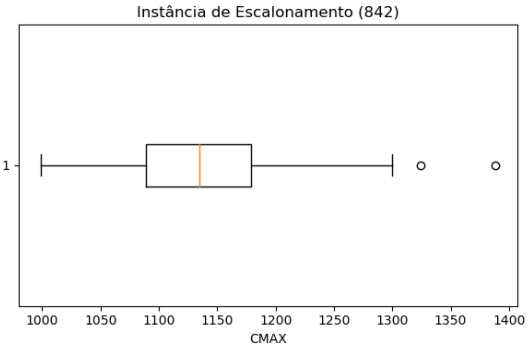

Instância de Escalonamento (661)
1
1000 1050 1100 1150 1200 1250 1300 1350 1400
CMAX

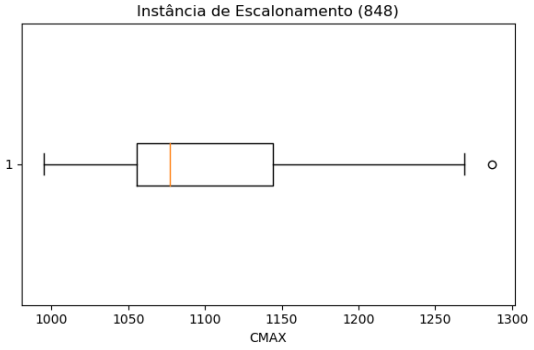

Instância de Escalonamento (666)
1
750 800 850 900 950 1000 1050 1100
CMAX

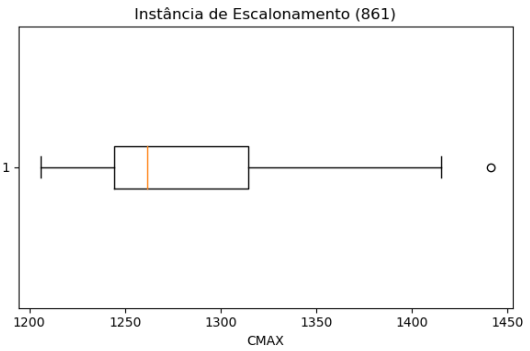

Instância de Escalonamento (784)
1
900 1000 1100 1200 1300
CMAX

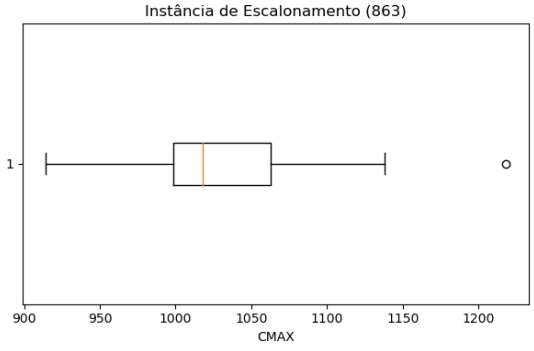

Instância de Escalonamento (826)
1
1200 1300 1400 1500 1600 1700 1800
CMAX

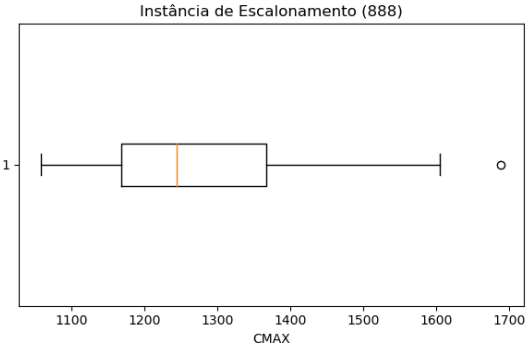

Instância de Escalonamento (827)
1
1200 1400 1600 1800 2000 2200 2400
CMAX

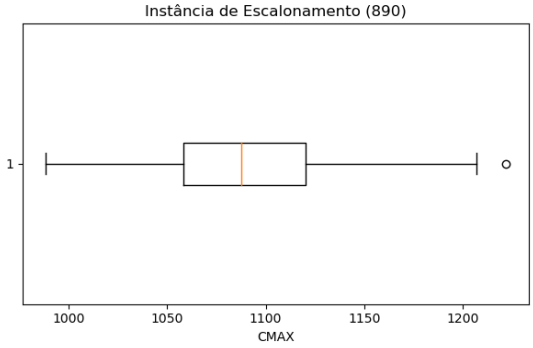

Instância de Escalonamento (842)
1
1000 1050 1100 1150 1200 1250 1300 1350 1400
CMAX

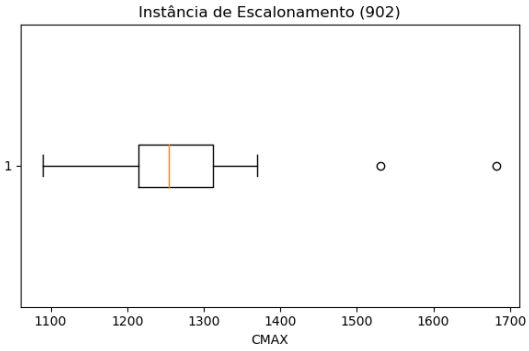

Instância de Escalonamento (848)
1
1000 1050 1100 1150 1200 1250 1300
CMAX

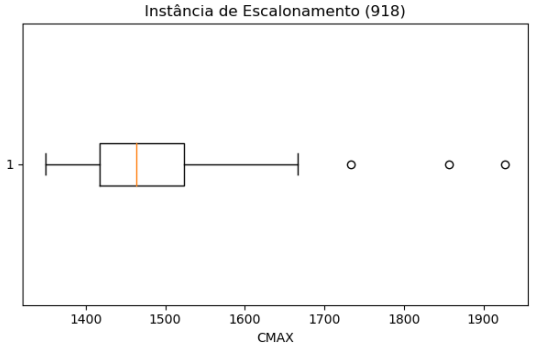

Instância de Escalonamento (861)
1
1200 1250 1300 1350 1400 1450
CMAX

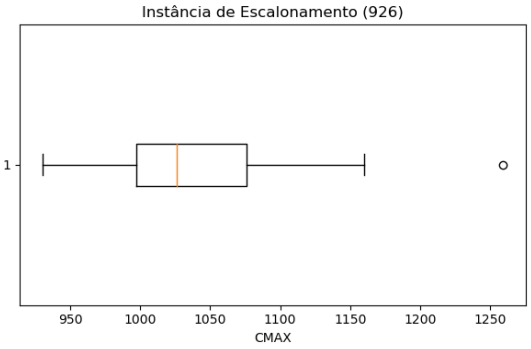

Instância de Escalonamento (863)
1
900 950 1000 1050 1100 1150 1200
CMAX

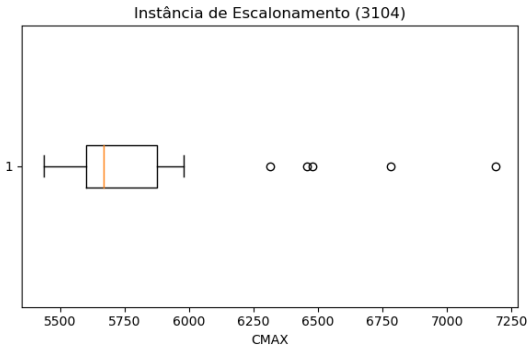

Instância de Escalonamento (888)
1
1100 1200 1300 1400 1500 1600 1700
CMAX

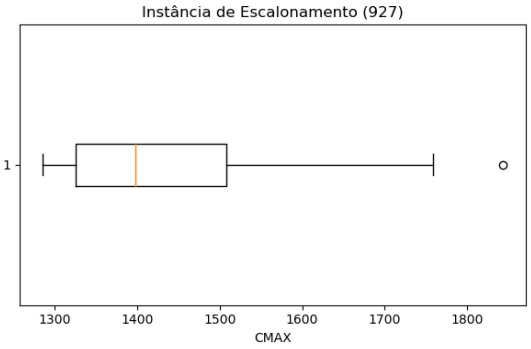

Instância de Escalonamento (890)
1
1000
1050
1100
1150
1200
CMAX

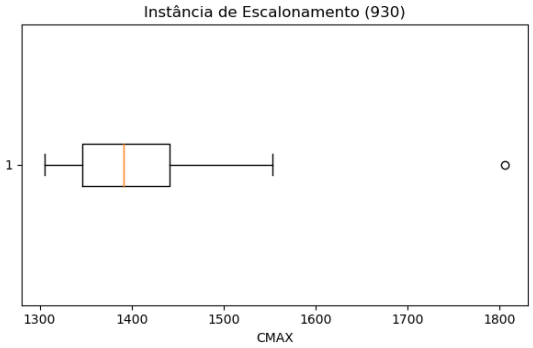

Instância de Escalonamento (902)
1
1100
1200
1300
1400
1500
1600
1700
CMAX

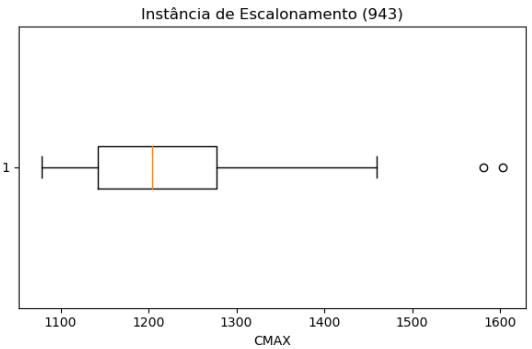

Instância de Escalonamento (918)
1
1400
1500
1600
1700
1800
1900
CMAX

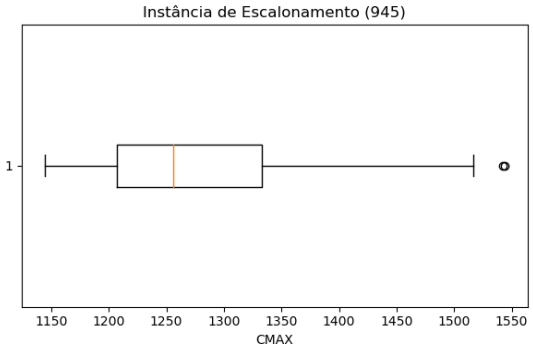

Instância de Escalonamento (926)
1
950
1000
1050
1100
1150
1200
1250
CMAX

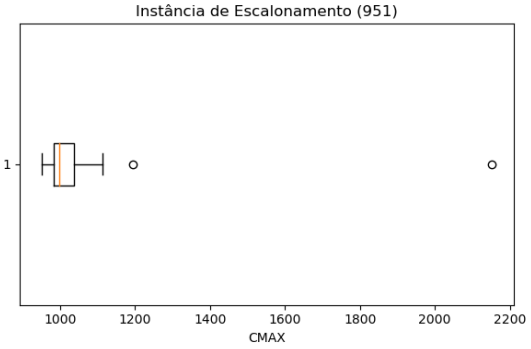

Instância de Escalonamento (3104)
1
5500
5750
6000
6250
6500
6750
7000
7250
CMAX

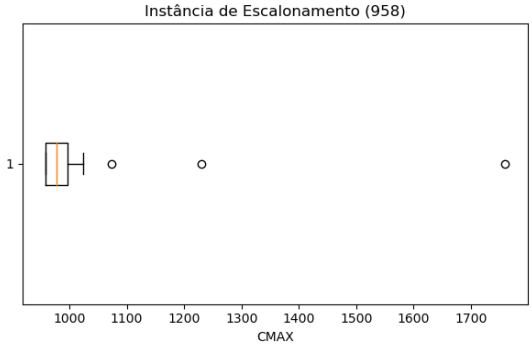

Instância de Escalonamento (927)
1
1300
1400
1500
1600
1700
1800
CMAX

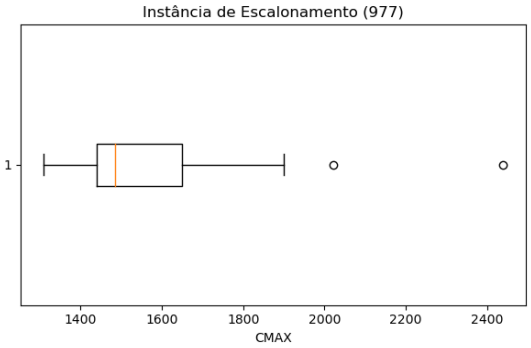

Instância de Escalonamento (930)
1
1300
1400
1500
1600
1700
1800
CMAX

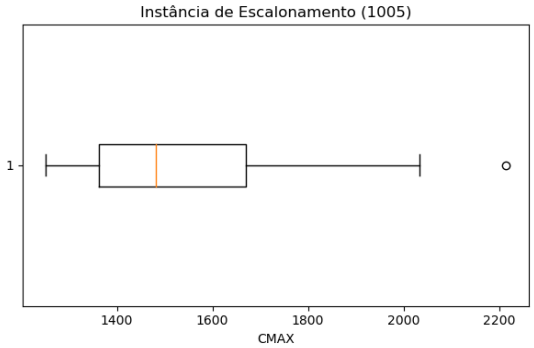

Instância de Escalonamento (943)
1
1100
1200
1300
1400
1500
1600
CMAX

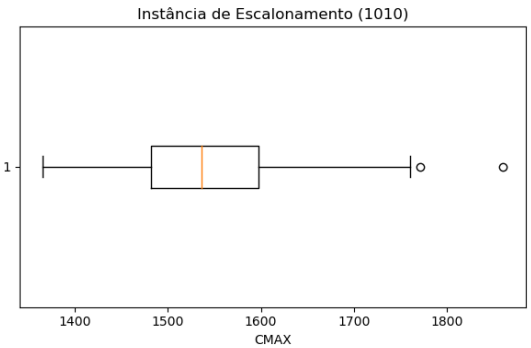

Instância de Escalonamento (945)
1
1150
1200
1250
1300
1350
1400
1450
1500
1550
CMAX

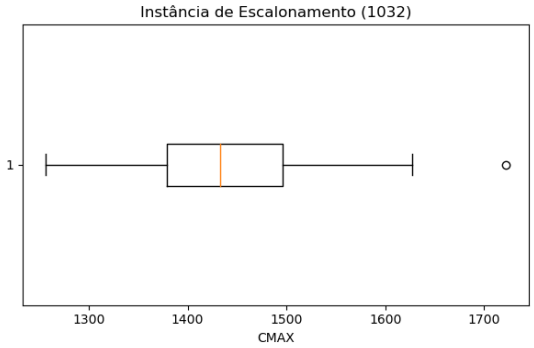

Instância de Escalonamento (951)
1
1000
1200
1400
1600
1800
2000
2200
CMAX

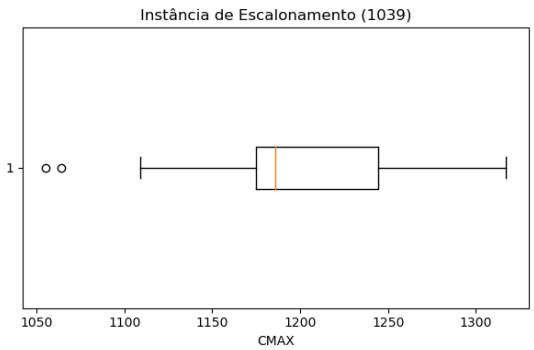

Instância de Escalonamento (958)
1
1000 1100 1200 1300 1400 1500 1600 1700
CMAX

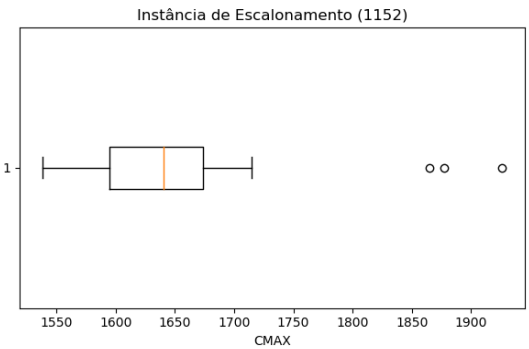

Instância de Escalonamento (977)
1
1400 1600 1800 2000 2200 2400
CMAX

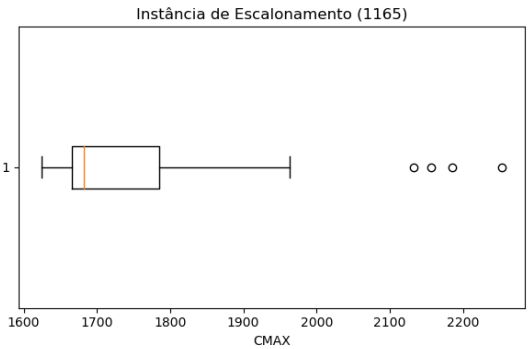

Instância de Escalonamento (1005)
1
1400 1600 1800 2000 2200
CMAX

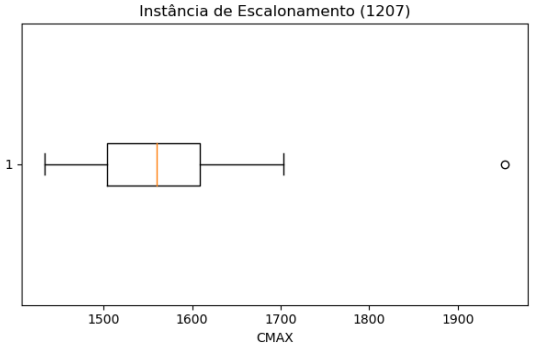

Instância de Escalonamento (1010)
1
1400 1500 1600 1700 1800
CMAX

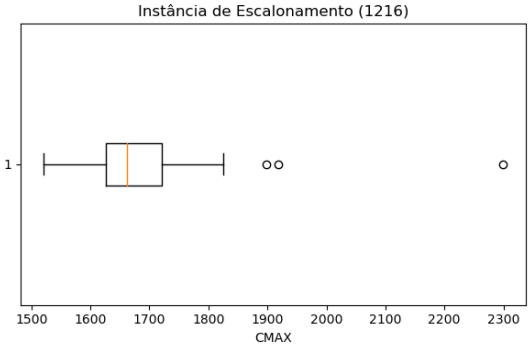

Instância de Escalonamento (1032)
1
1300 1400 1500 1600 1700
CMAX

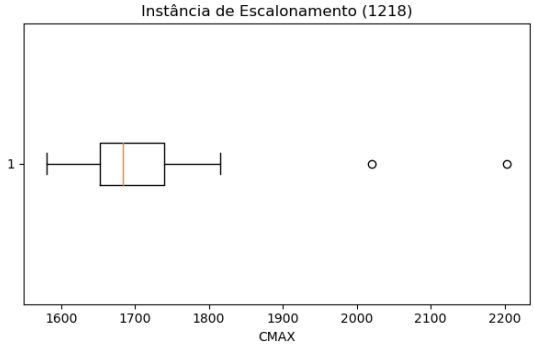

Instância de Escalonamento (1039)
1
1050 1100 1150 1200 1250 1300
CMAX

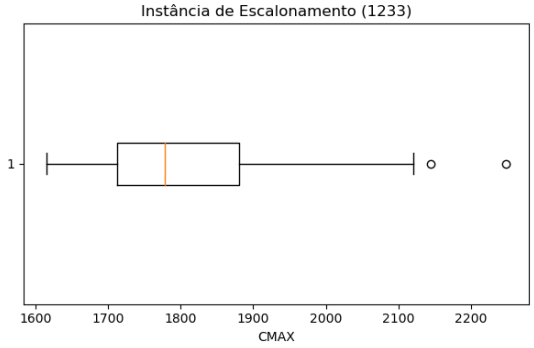

Instância de Escalonamento (1152)
1
1550 1600 1650 1700 1750 1800 1850 1900
CMAX

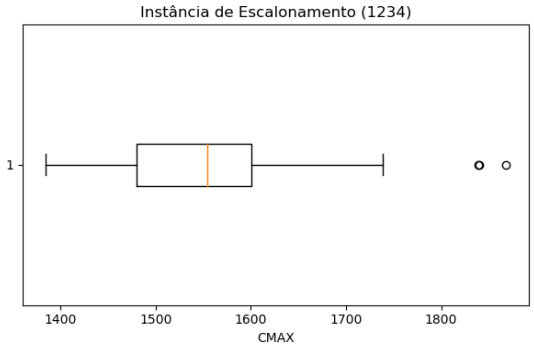

Instância de Escalonamento (1165)
1
1600 1700 1800 1900 2000 2100 2200
CMAX

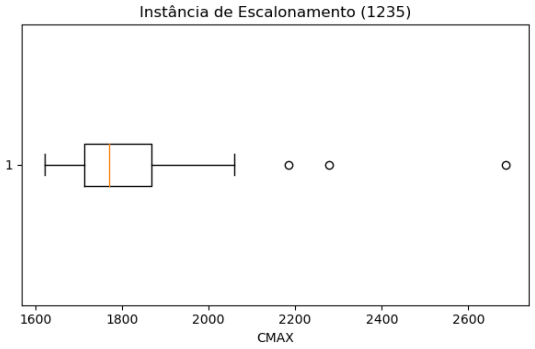

Instância de Escalonamento (1207)
1
1500 1600 1700 1800 1900
CMAX

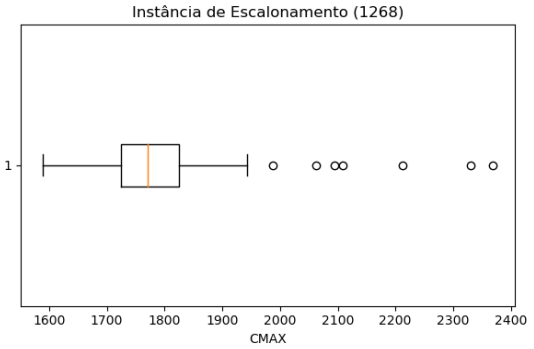

Instância de Escalonamento (1216)
1
1500 1600 1700 1800 1900 2000 2100 2200 2300
CMAX

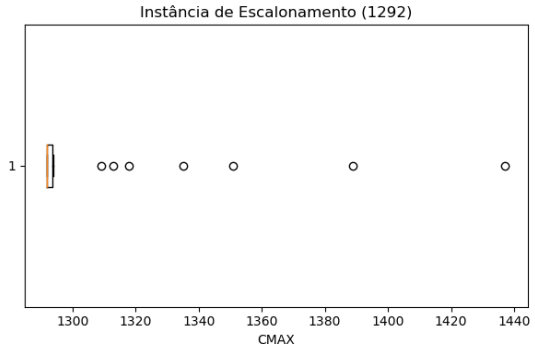

Instância de Escalonamento (1218)
1
1600
1700
1800
1900
2000
2100
2200
CMAX

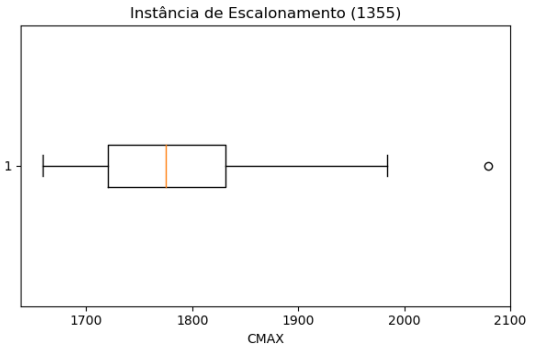

Instância de Escalonamento (1233)
1
1600
1700
1800
1900
2000
2100
2200
CMAX

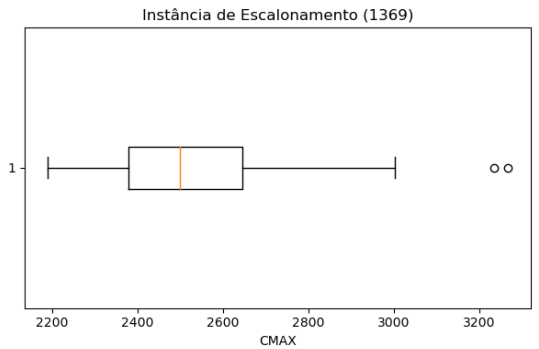

Instância de Escalonamento (1234)
1
1400
1500
1600
1700
1800
CMAX

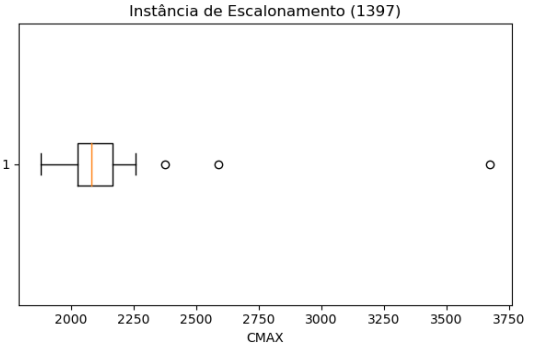

Instância de Escalonamento (1235)
1
1600
1800
2000
2200
2400
2600
CMAX

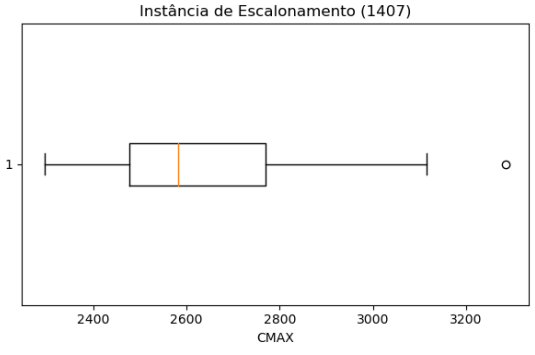

Instância de Escalonamento (1268)
1
1600
1700
1800
1900
2000
2100
2200
2300
2400
CMAX

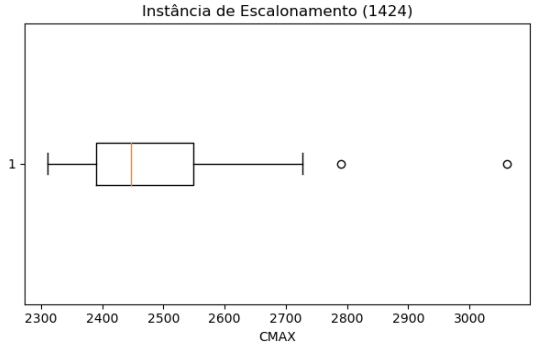

Instância de Escalonamento (1292)
1
1300
1320
1340
1360
1380
1400
1420
1440
CMAX

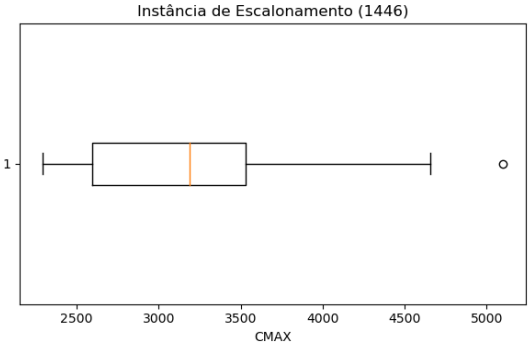

Instância de Escalonamento (1355)
1
1700
1800
1900
2000
2100
CMAX

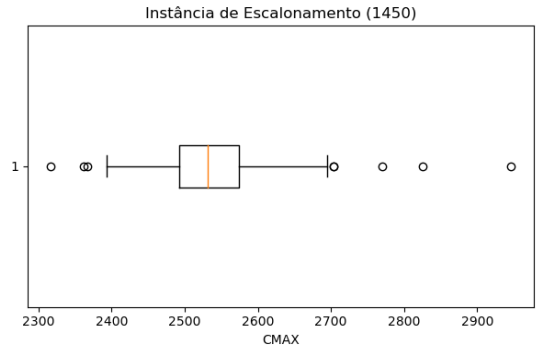

Instância de Escalonamento (1369)
1
2200
2400
2600
2800
3000
3200
CMAX

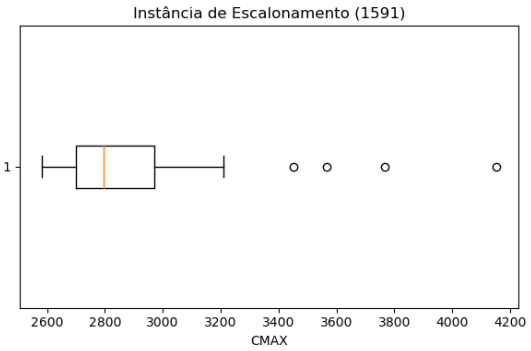

Instância de Escalonamento (1397)
1
2000
2250
2500
2750
3000
3250
3500
3750
CMAX

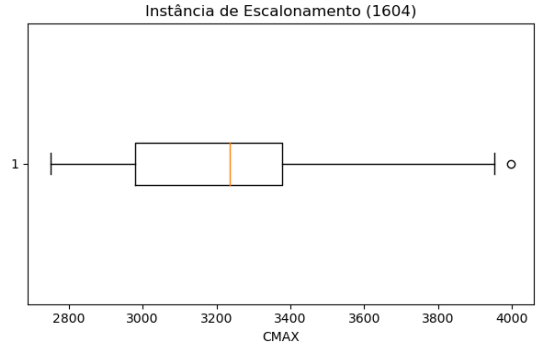

Instância de Escalonamento (1407)
1
2400
2600
2800
3000
3200
CMAX

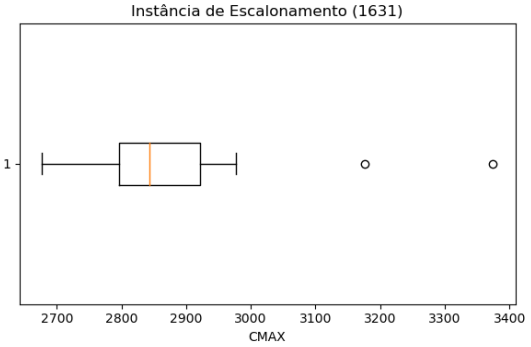

Instância de Escalonamento (1424)
1
2300 2400 2500 2600 2700 2800 2900 3000
CMAX

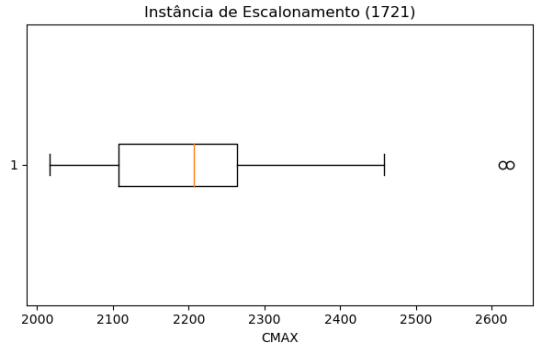

Instância de Escalonamento (1446)
1
2500 3000 3500 4000 4500 5000
CMAX

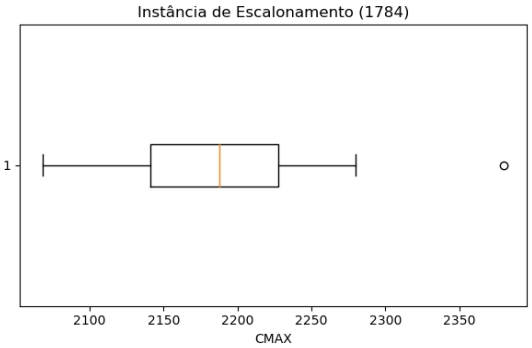

Instância de Escalonamento (1450)
1
2300 2400 2500 2600 2700 2800 2900
CMAX

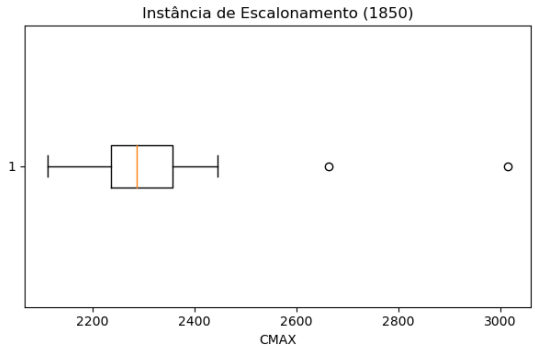

Instância de Escalonamento (1591)
1
2600 2800 3000 3200 3400 3600 3800 4000 4200
CMAX

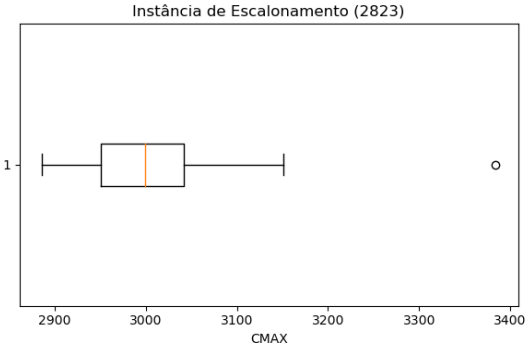

Instância de Escalonamento (1604)
1
2800 3000 3200 3400 3600 3800 4000
CMAX

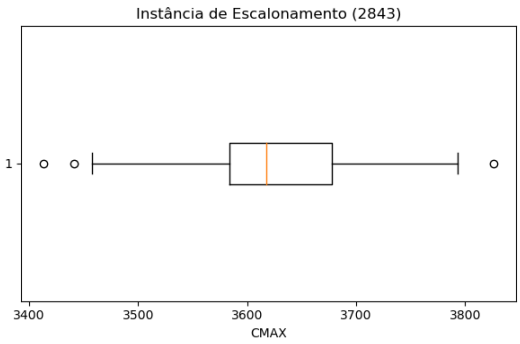

Instância de Escalonamento (1631)
1
2700 2800 2900 3000 3100 3200 3300 3400
CMAX

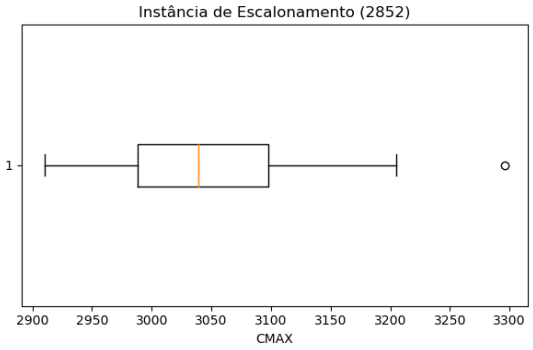

Instância de Escalonamento (1721)
1
2000 2100 2200 2300 2400 2500 2600
CMAX

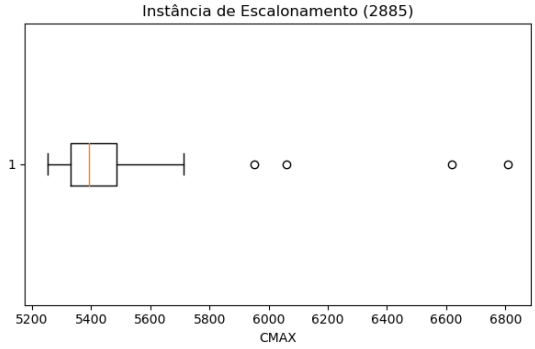

Instância de Escalonamento (1784)
1
2100 2150 2200 2250 2300 2350
CMAX

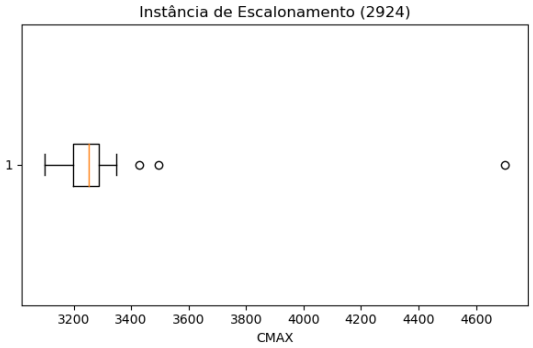

Instância de Escalonamento (1850)
1
2200 2400 2600 2800 3000
CMAX

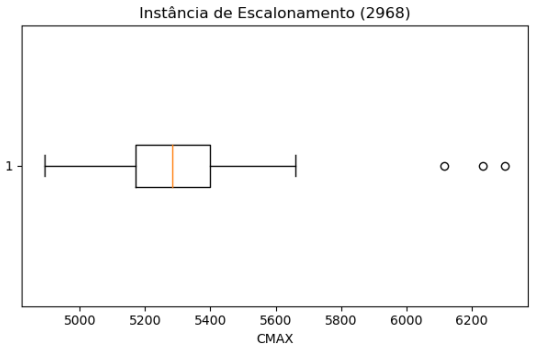

Instância de Escalonamento (2823)
1
2900 3000 3100 3200 3300 3400
CMAX

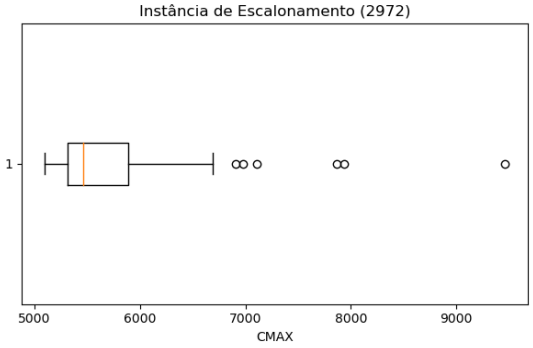

Instância de Escalonamento (2843)
1
3400
3500
3600
3700
3800
CMAX

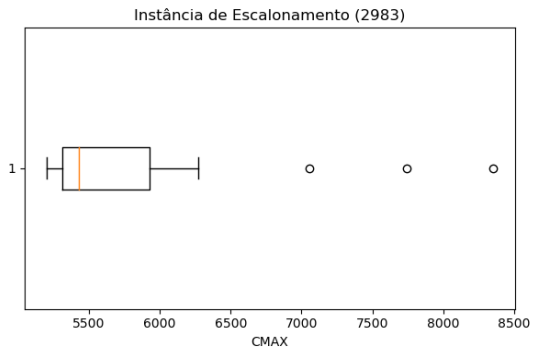

Instância de Escalonamento (2852)
1
2900
2950
3000
3050
3100
3150
3200
3250
3300
CMAX


Instância de Escalonamento (2885)
1
5200
5400
5600
5800
6000
6200
6400
6600
6800
CMAX


Instância de Escalonamento (2924)
1
3200
3400
3600
3800
4000
4200
4400
4600
CMAX


Instância de Escalonamento (2968)
1
5000
5200
5400
5600
5800
6000
6200
CMAX


Instância de Escalonamento (2972)
1
5000
6000
7000
8000
9000
CMAX


Instância de Escalonamento (2983)
1
5500
6000
6500
7000
7500
8000
8500
CMAX

Anexo B

As técnicas de Ensemble, frequentemente utilizadas na Machine Learning, combinam as previsões de vários modelos base com o intuito de obter um desempenho superior ao que seria possível com apenas um único modelo. Pressupõe-se que diferentes modelos podem captar diversos padrões e, ao combinar as suas previsões, o modelo de Ensemble pode compensar as fraquezas dos modelos individuais e produzir um desempenho melhor.

A Ensemble Voting é uma abordagem comum na aprendizagem de Ensemble em que a previsão final é efetuada através da combinação das previsões de vários modelos individuais. Nesta técnica, cada modelo individual do conjunto realiza, de forma independente, previsões sobre os dados de entrada e, a previsão final é determinada pela combinação das previsões destes modelos através de um esquema de votação.

No modelo preditivo desenvolvido apenas foram consideradas as técnicas Ensemble Voting de votação *hard* e votação *soft*. A votação *hard* envolve o simples ato de contar as previsões feitas por cada modelo base e selecionar a classe que recebe mais "votos" como previsão final. A votação *soft* tem em conta as pontuações de probabilidade de cada modelo de base para cada classe e calcula a média ponderada das probabilidades para realizar a previsão final.

O modelo preditivo de Ensemble Voting desenvolvido baseia-se na combinação de apenas duas técnicas de Machine Learning. Como o algoritmo Random Forest apresenta os melhores resultados de desempenho, optou-se por realizar combinações entre este algoritmo e cada uma das restantes técnicas, correspondendo a quatro testes distintos. Para além disso, na construção deste modelo, foi utilizado o melhor modelo individual de cada técnica de Machine Learning. Isto é, para cada algoritmo, recorreu-se à utilização do melhor conjunto de hiperparâmetros obtido através da implementação da técnica GridSearchCV. A aplicação da técnica GridSearchCV encontra-se descrita na secção 4.2.

Após a elaboração e implementação dos modelos preditivos de Ensemble Voting, verifica-se que dois modelos exibem os melhores resultados nos indicadores de desempenho, designadamente os modelos com a combinação Random Forest-Árvore de Decisão e Random Forest-Support Vector Machines. Na Tabela 26 é apresentado os resultados obtidos nas métricas de avaliação dos modelos Random Forest-Árvore de Decisão e Random Forest-Support Vector Machines.

Tabela 26 - Resultados das métricas de avaliação - Modelo de ensemble voting

Combinação	Tipo de votação	Accuracy	F1 Score ponderado	Precision	Recall	AUC
Random Forest – Árvore de Decisão	Hard	0,66	0,6627	0,6884	0,66	0,8906
	Soft	0,6865	0,6869	0,6997	0,6865	0,8906
Random Forest – Support Vector Machines	Hard	0,6832	0,6843	0,6943	0,6832	0,89
	Soft	0,6633	0,6609	0,6721	0,6633	0,8899

Analisando a Tabela 26, verifica-se que a combinação Random Forest-Árvore de Decisão obteve melhores resultados na votação *soft*, apresentando valores de Accuracy, F1 Score ponderado, Precision, Recall e AUC de 0,6865, 0,6869, 0,6997, 0,6865 e 0,8906, respetivamente. A votação *hard*, nesta combinação, embora apresente valores ligeiramente inferiores, exibe um desempenho consistente, com uma Precision ainda elevada de 0,6884. Além disso, é importante realçar que ambas as votações demonstram o mesmo valor de AUC (0.8906).

Relativamente à combinação Random Forest-Support Vector Machines, esta apresenta melhores indicadores de desempenho na votação *hard*, com valores de 0,6832, 0,6843, 0,6943, 0,6832, 0,89 para Accuracy, F1 Score ponderado, Precision, Recall e AUC, respetivamente. Por outro lado, a votação *soft*, registou valores mais baixos em todas as métricas de avaliação.

Comparando os melhores resultados de cada modelo, conclui-se que a combinação Random Forest-Árvore de Decisão demonstra um desempenho ligeiramente superior do que a combinação Random Forest-Support Vector Machines, apresentando os valores mais elevados nas métricas de avaliação. Contudo, apesar dos resultados satisfatórios, a técnica Ensemble Voting não foi implementada no modelo preditivo, uma vez que as melhorias observadas não se relevaram suficientemente significativas para justificar a sua aplicação.