



Aplicação de Modelos de Linguagem de Grande Escala a problemas de Escalonamento das Operações

JOÃO PEDRO FERNANDES MARQUES

Setembro de 2025

**Aplicação de Modelos de Linguagem de Grande
Escala a problemas de Escalonamento das
Operações**

João Pedro Fernandes Marques

**Dissertação para obtenção do Grau de Mestre em
Engenharia Mecânica, Área de Especialização em
Gestão Industrial**

Orientador: André Borges Guimarães Serra e Santos

Júri:

Presidente:

Vogais:

Agradecimentos

Este trabalho representa a conclusão de um período da minha vida de muito esforço e sacrifício, dando por fim ao meu percurso acadêmico. A realização deste trabalho não seria possível sem o apoio, incentivo e orientação das pessoas à minha volta, às quais deixo o meu sincero agradecimento.

Quero agradecer antes de mais ao meu orientador, engenheiro André Serra e Santos, pela atenção e disponibilidade constante ao longo destes meses, destacando a proatividade em garantir um acompanhamento regular através de reuniões, e os conselhos fornecidos que ajudaram a progredir o trabalho.

À minha família, os meus pais e o meu irmão, que sempre me apoiaram incondicionalmente em todas as decisões que tomei, expresso a minha profunda gratidão pela paciência que tiveram comigo e pela constante fonte de motivação que me deram ao longo do meu percurso académico.

Não poderia deixar de referir os meus amigos pelo encorajamento, pela companhia e pelos momentos de partilha que tornaram a conclusão desta etapa, e todo o percurso, mais gratificante.

Resumo

Os problemas de Escalonamento das Operações são particularmente relevantes em ambientes industriais, e fundamentais para tornar a produção eficiente e cumprir as datas estipuladas com os clientes. Em ambientes de grande dimensão, encontrar soluções ótimas pode tornar-se um desafio que requer tempo e recursos computacionais significativos. No entanto, o rápido avanço da Inteligência Artificial tem sido uma enorme ajuda no ramo da gestão industrial. Um dos principais campos de IA a ser desenvolvido nos últimos anos é o de IA Generativa. Neste trabalho procura-se explorar as capacidades de três Modelos de Linguagem de Grande Escala distintos, DeepSeek R1, GPT o4-mini e GPT o3, através da resolução de um conjunto de 19 instâncias em ambiente *Job Shop* de diferente complexidade. Os modelos demonstraram variabilidade no desempenho, reforçando a importância do aperfeiçoamento dos *prompts* introduzidos através de engenharia de *prompts*. Para ajudar a estabilizar os resultados, foi desenvolvido um manual de *prompting* com foco na resolução deste tipo de problemas, independentemente do modelo a ser utilizado. Os dois primeiros modelos não conseguiram gerar soluções para mais de metade das instâncias, sendo observada uma dificuldade em respeitar as restrições impostas nos problemas mais complexos. O modelo DeepSeek R1, em especial, apenas conseguiu gerar solução para 6 das 19 instâncias e apresentou a pior eficiência entre os três modelos, demorando por vezes mais de 10 minutos a resolver problemas de menor complexidade. Já o GPT o3, disponível numa plataforma de subscrição, demonstrou um desempenho superior, gerando soluções ótimas ou próximas do ótimo para todas as instâncias, embora ainda tenha precisado de um tempo de processamento não negligenciável para gerar as respostas. Concluiu-se que, com o cuidado adequado e preparação prévia dos *prompts*, este modelo representa um recurso relevante na otimização dos problemas de Escalonamento das Operações. Adicionalmente, como o modelo GPT o4-mini apresentou um tempo de processamento das respostas muito inferior aos outros dois, pode constituir-se como uma alternativa eficiente para a otimização de problemas menos complexos.

Palavras-chave: Escalonamento das Operações, Inteligência Artificial, Modelos de Linguagem de Grande Escala, ChatGPT, DeepSeek, *Prompting*

Abstract

Operations Scheduling problems are particularly relevant in industrial environments and are fundamental to ensuring efficient production and meeting client deadlines. In large-scale environments, finding optimal solutions can be a challenge that requires significant time and computational resources. However, the rapid advancement of Artificial Intelligence has been a great help in the field of industrial management. One of the main areas of AI being developed in recent years is Generative AI. This work seeks to explore the capabilities of three distinct Large Language Models, DeepSeek R1, GPT o4-mini and GPT o3, by solving a set of 19 instances of varying complexity in Job Shop environment. The models showed some variability in their performance, reinforcing the importance of improving the prompts introduced, through the implementation of prompt engineering. To help stabilize the results, a manual of prompting was developed focusing on solving this type of problem, regardless of the model being used. The first two models were unable to generate solutions for more than half of the instances, with difficulty observed in respecting the constraints imposed on the most complex problems. The DeepSeek R1 model was only able to generate solutions for 6 of the 19 instances and had the worst efficiency among the three models, sometimes taking more than 10 minutes to solve less complex problems. GPT o3, available on a subscription platform, demonstrated superior performance, generating optimal or near-optimal solutions for all instances, although it still required a significant amount of processing time to generate the responses. It was concluded that, with proper care and prior preparation of the prompts, this model represents a relevant resource in the optimization of Operations Scheduling problems. Additionally, as the GPT o4-mini model had a much shorter response processing time than the other two, it may be an efficient alternative for optimizing less complex problems.

KEYWORDS: Operations Scheduling, Artificial Intelligence, Large Language Models, ChatGPT, DeepSeek, Prompting

Índice

Lista de Figuras.....	xi
Lista de Tabelas.....	xiii
Acrónimos e Símbolos.....	xv
1. Introdução.....	1
1.1. Contextualização	1
1.2. Objetivos	2
1.3. Metodologia	2
1.4. Estrutura do Trabalho	3
2. Revisão Bibliográfica	5
2.1. Escalonamento das Operações	5
2.2. Definição	6
2.3. Fases do Escalonamento	7
2.3.1. Problema de Afetação	7
2.3.2. Problema da Sequenciação.....	8
2.3.3. Problema de Calendarização	8
2.4. Representação.....	9
2.5. Ambiente de produção	10
2.5.1. Máquina Única.....	10
2.5.2. Máquinas Paralelas.....	11
2.5.3. Flow Shop.....	11
2.5.4. Job Shop.....	12
2.5.5. Open Shop.....	12
2.6. Natureza do Escalonamento	12
2.7. Inteligência Artificial.....	13
2.8. Machine Learning.....	14
2.8.1. Deep Learning	17
2.9. Large Language Models.....	18
2.9.1. ChatGPT	20
2.9.2. Copilot e Gemini	22
2.9.3. DeepSeek	24
2.10. Prompting.....	24
2.11. Análise Bibliométrica.....	27
2.11.1. Revisão do estado da arte	30
2.12. Análise crítica da revisão bibliográfica	34
3. Métodos e Aplicações.....	35

3.1. Problemas de Escalonamento das Operações	35
3.2. Métodos de otimização.....	38
3.3. Preparação dos LLMs	44
3.4. Aplicação dos LLMs na resolução de problemas de Escalonamento	48
3.5. Conclusão	70
4. Resultados e Discussão	71
4.1. Apresentação dos Resultados	71
4.2. Estatística Descritiva	74
4.3. Inferência Estatística	79
4.4. Discussão de Resultados	83
5. Conclusão	85
5.1. Conclusões Finais	85
5.2. Limitações e Trabalhos Futuros	87
Referências.....	89
Declaração de Integridade	97
Apêndice A	99

Lista de Figuras

Figura 1 - Representação do Problema de Afetação (adaptado de [7])	7
Figura 2 - Representação do Problema de Sequenciação (adaptado de [7]).....	8
Figura 3 - Representação do Problema de Calendarização	8
Figura 4 - Evolução da Inteligência Artificial (adaptado de [30])	14
Figura 5 - Comparação de Deep Learning com Inteligência Artificial e Machine Learning (Adaptado de [32]).....	17
Figura 6 - Exemplo da estrutura de uma Deep Neural Network (DNN) [59]	18
Figura 7 - Tendência de artigos publicados por ano sobre "Large Language Models" [61]	19
Figura 8 - Tempo que plataformas online demoraram a atingir 1 milhão de utilizadores (adaptado de [66])	20
Figura 9 - Uso de ChatGPT por área profissional [67].....	21
Figura 10 - Desempenho de GPT-3.5, GPT-4 e GPT-4 (sem visão) em vários testes académicos	22
Figura 11 - Diferença entre prompts simples e complexos no ChatGPT	25
Figura 12 - Distribuição temporal de publicações entre 2012 e 2024	28
Figura 13 - Distribuição de publicações por áreas de investigação	28
Figura 14 - Mapa de co-ocorrência de palavras-chave	29
Figura 15 - Relação Tempo Máximo de Conclusão / Consumo Energético	31
Figura 16 - Curvas de convergência do algoritmo.....	32
Figura 17 - Gráfico de Gantt do problema exemplo em Job Shop.....	37
Figura 18 - Grafo Disjuntivo do problema exemplo em Job Shop	38
Figura 19 - Resposta de ChatGPT sobre aplicação de MILP	39
Figura 20 - Pensamento de ChatGPT sobre aplicação de CP	40
Figura 21 - Exemplo de um algoritmo Branch and Bound (adaptado de [104])	41
Figura 22 - Representação do algoritmo Simplex	42
Figura 23 - Gráfico de Gantt para a primeira tentativa de solução de uma instância 5Jx3M....	47
Figura 24 - Gráfico de Gantt para a segunda tentativa de solução de uma instância 5Jx3M....	47
Figura 25 - Gráfico de Gantt representativo da solução ótima do problema [3Jx3M]	60
Figura 26 - Primeiro gráfico Gantt gerado pelo GPT o4-mini para [10Jx3M].....	63
Figura 27 - Segundo gráfico Gantt gerado pelo GPT o4-mini para [10Jx3M]	65
Figura 28 - Gráfico Gantt gerado pelo DeepSeek R1 para [10Jx3M]	66
Figura 29 - Gráfico Gantt gerado pelo GPT o3 para [10Jx3M]	67
Figura 30 - Número de tentativas necessárias para cada modelo.....	74
Figura 31 - Relação Makespan ótimo com Makespan gerado pelo DeepSeek R1.....	75
Figura 32 - Relação Makespan ótimo com Makespan gerado pelo GPT o4-mini	75
Figura 33 - Relação Makespan ótimo com Makespan gerado pelo GPT o3	76
Figura 34 - Variação da Proximidade com o Número de Máquinas para os 3 modelos.....	78
Figura 35 - Variação da Eficiência com o Número de Máquinas para os 3 modelos	78

Lista de Tabelas

Tabela 1 - Exemplos de aplicações dos diferentes tipos de aprendizagem	15
Tabela 2 - Exatidão dos LLMs em sete tarefas de diferentes categorias (adaptado de [74])	23
Tabela 3 - Pesquisa de palavras-chave no Web of Science.....	27
Tabela 4 - Resultados médios para TFIDF e LLM.....	30
Tabela 5 - Resultados médios para programação inteira com TFIDF e LLM.....	30
Tabela 6 - Outros Casos de Estudo Relevantes.....	33
Tabela 7 - Tempos de processamento de um problema exemplo em Job Shop com 4 tarefas e 3 máquinas.....	37
Tabela 8 - Evolução dos prompts	45
Tabela 9 - 1º prompt: conhecimento base sobre Escalonamento das Operações.....	48
Tabela 10 - 2º prompt: conhecimento sobre JSSP	50
Tabela 11 - 3º prompt: conhecimento sobre métodos de otimização	51
Tabela 12 - Resolução do problema de Job Shop [3Jx3M] pelo GPT o4-mini.....	55
Tabela 13 - Resolução do problema de Job Shop [3Jx3M] pelo DeepSeek R1	57
Tabela 14 - Lista de instâncias usadas para teste nos LLMs	61
Tabela 15 - Primeiro programa gerado pelo GPT o4-mini para [10Jx3M]	62
Tabela 16 - Segundo programa gerado pelo GPT o4-mini para [10Jx3M]	64
Tabela 17 - Programa gerado pelo GPT o3 para [10Jx3M]	68
Tabela 18 - Resultados dos testes às 19 instâncias.....	72
Tabela 19 - Análise Descritiva de Proximidade para os 3 modelos.....	77
Tabela 20 - Análise Descritiva de Tempo de Processamento para os 3 modelos.....	77
Tabela 21 - Análise Descritiva de Eficiência para os 3 modelos.....	77
Tabela 22 - Análise Descritiva de Número de Tentativas para os 3 modelos	77
Tabela 23 - Resultados do teste de Normalidade (Shapiro-Wilk).....	80
Tabela 24 - Resultados do teste Kruskal-Wallis	81
Tabela 25 - Resultados do teste Mann-Whitney para (DeepSeek R1 - GPT o4-mini).....	82
Tabela 26 - Resultados do teste Mann-Whitney para (DeepSeek R1 - GPT o3)	82
Tabela 27 - Resultados do teste Mann-Whitney para (GPT o4-mini - GPT o3).....	83

Acrónimos e Símbolos

Lista de Acrónimos

AI	<i>Artificial Intelligence</i>
ANN	<i>Artificial Neural Networks</i>
BERT	<i>Bidirecional Encoder Representations from Transformers</i>
CoT	<i>Chain of Thought</i>
CP	<i>Constraint Programming</i>
CPS	<i>Cyber Physical System</i>
DBN	<i>Deep Belief Network</i>
DL	<i>Deep Learning</i>
DNN	<i>Deep Neural Network</i>
EDD	<i>Earliest Due Date</i>
EFT	<i>Earliest Finish Time</i>
EST	<i>Earliest Start Time</i>
FCFS	<i>First Come First Served</i>
FIFO	<i>First In First Out</i>
GA	<i>Genetic Algorithm</i>
GAN	<i>Generative Adversarial Network</i>
GPT	<i>Generative Pre-Trained Transformer</i>
IA	Inteligência Artificial
IBM	<i>International Business Machines</i>
ILOG	<i>Intelligence Logiciel</i>
ISEP	Instituto Superior de Engenharia do Porto
LLM	<i>Large Language Model</i>
LPT	<i>Longest Processing Time</i>
MILP	<i>Mixed Integer Linear Programming</i>
ML	<i>Machine Learning</i>
MoE	<i>Mixture of Experts</i>
NEH	<i>Nawaz-Enscore-Ham</i>
NSGA	<i>Non-dominated Sorting Genetic Algorithm</i>
PLN	Processamento de Linguagem Natural

PPCP	Planeamento, Programação e Controlo da Produção
P.Porto	Instituto Politécnico do Porto
RL	<i>Reinforcement Learning</i>
SA	<i>Simulated Annealing</i>
SGS	<i>Serial Schedule Generation Scheme</i>
SL	<i>Supervised Learning</i>
SPT	<i>Shortest Processing Time</i>
TFIDF	<i>Term Frequency–Inverse Document Frequency</i>
TS	<i>Tabu Search</i>
UL	<i>Unsupervised Learning</i>

1. Introdução

Neste capítulo, será feita uma introdução ao tema do trabalho, detalhando o contexto e enquadramento do trabalho, os objetivos propostos, a metodologia adotada para os alcançar e, finalmente, a estrutura do documento.

1.1. Contextualização

Em ambientes empresariais marcados por um dinamismo e competitividade crescentes, as organizações são desafiadas a manterem-se relevantes, responder à procura em constante evolução e adaptar-se rapidamente às exigências do mercado. Este dinamismo decorre de fatores como a complexidade dos sistemas industriais, que envolvem múltiplas máquinas e restrições operacionais, possíveis interrupções na produção, tempos de espera entre tarefas e a necessidade de reconfiguração rápida das linhas de produção.

Um ambiente tão incerto torna o Escalonamento das Operações um processo essencial, mas altamente desafiadora. Esta é uma área fundamental da gestão industrial, desempenhando um papel crucial na eficiência e eficácia das linhas de produção, que envolve decisões táticas para alocar recursos, definir sequências de tarefas e estabelecer calendários de execução. Através de uma gestão eficaz do escalonamento, é possível minimizar custos, melhorar a utilização dos recursos e aumentar a capacidade de resposta às variações da procura.

Paralelamente, o campo da Inteligência Artificial (IA) tem demonstrado um potencial cada vez maior para lidar com problemas complexos. Técnicas de Machine Learning, como os *Large Language Models* (LLMs) têm revolucionado a forma como são analisados e resolvidos problemas. Estes modelos de linguagem destacam-se pela sua capacidade de compreender linguagem natural e realizar tarefas baseadas em padrões e inferências. A sua utilização em problemas técnicos, como o Escalonamento de Operações, é promissora, embora ainda pouco explorada de forma sistemática.

Este trabalho insere-se nesta interseção, explorando o potencial dos LLMs na resolução de problemas de Escalonamento. Pretende-se investigar como estas ferramentas podem ser aplicadas para enfrentar problemas específicos de Escalonamento, avaliando a sua eficácia no planeamento e gestão das operações em ambientes dinâmicos.

1.2. Objetivos

Neste trabalho, procura-se avaliar as capacidades de IA Generativa na resolução de problemas de Escalonamento das Operações em *Job Shop*. Serão utilizados Modelos de Linguagem de Grande Escala muito relevantes atualmente, para observar o comportamento deste domínio da Inteligência Artificial a lidar com problemas de diferentes graus de complexidade. Inicialmente considerou-se essencial realizar uma investigação sobre os temas de Escalonamento das Operações e de Inteligência Artificial para enquadrar o restante trabalho. O estudo centra-se na avaliação da capacidade dos modelos apresentarem soluções de forma eficiente, permitindo explorar o poder de raciocínio dos mesmos. Pretende-se também assistir o utilizador na sua interação com os LLMs, para tal é apresentado um manual de *prompting* de forma a encaminhar o modelo mais facilmente à compreensão correta do problema de Escalonamento e a uma resposta mais apropriada.

De modo a chegar a uma conclusão acerca do desempenho dos Modelos de Linguagem de Grande Escala, será necessário realizar as seguintes etapas:

- Revisão literária da área de Escalonamento das Operações, formas de representação dos problemas, tipos de ambientes de produção e métodos de otimização;
- Revisão literária da área de Inteligência Artificial, com foco em Modelos de Linguagem de Grande Escala;
- Escolha de Modelos de Linguagem de Grande Escala a utilizar para este estudo;
- Aperfeiçoamento dos *prompts* para obter respostas mais apropriadas e desenvolvimento de um manual de *prompting*;
- Fase de teste dos LLMs selecionados através de um conjunto de instâncias de Escalonamento em *Job Shop*;
- Realização de uma análise estatística dos resultados obtidos, de forma a comparar o desempenho dos LLMs;

1.3. Metodologia

A abordagem metodológica adotada neste trabalho é de carácter exploratório e dedutivo, isto é, parte de uma análise prévia dos conceitos teóricos e de problemas abordados em artigos académicos para subsequente avaliação e aplicação prática.

Seguindo esta metodologia, foram realizadas várias etapas, delineadas da seguinte forma:

- Pesquisa preliminar breve e exploratória sobre a aplicação de LLMs para resolução de problemas de Escalonamento, servindo como base para definir os objetivos e estruturar o trabalho;
- Revisão bibliográfica, onde foram abordados conceitos teóricos fundamentais sobre Escalonamento das Operações, Inteligência Artificial, Machine Learning e LLMs com o objetivo de obter uma base teórica essencial para o tema do trabalho. Para tal foram

encontrados artigos acadêmicos e livros relevantes a este estudo, usados como referência para formular conclusões.

- Análise Bibliométrica para investigar detalhadamente os casos de estudo mais recentes e relevantes ao tema do trabalho, mapeando o estado da arte.

1.4. Estrutura do Trabalho

Este documento está organizado em cinco capítulos:

- Capítulo 1 - Introdução: é apresentado o contexto geral do trabalho, estabelecido os objetivos e a metodologia a seguir para os atingir, e por último a estrutura do trabalho dividida por capítulos.
- Capítulo 2 - Revisão Bibliográfica: em primeiro lugar, é feito um estudo de conceitos-chave sobre Escalonamento das Operações, apresentando as três fases possíveis de um problema, a Afetação, a Sequenciação e a Calendarização, bem como os diferentes cenários em oficina de trabalho. É também abordado o tema de Inteligência Artificial, Machine Learning e Deep Learning, e ainda Large Language Models, com especial foco no ChatGPT. Para além disso, este capítulo é complementado por uma análise bibliométrica, que ajuda a conectar os dois grandes tópicos do trabalho, suportada pelo estudo de casos focados na resolução de problemas de Escalonamento através de LLMs.
- Capítulo 3 – Métodos e Aplicações: aprofunda-se a compreensão dos problemas de Escalonamento das Operações, em específico de problemas *Job Shop*, apresentando a formulação matemática associada ao problema, assim como possíveis representações da solução. São também referidos os métodos de otimização utilizados para gerar calendários das operações. De seguida é abordada a fase de preparação dos LLMs, em que se aplicou técnicas de engenharia de *prompts* para melhorar a interação com os modelos. Finalmente são apresentadas algumas das resoluções dos problemas, através dos LLMs selecionados.
- Capítulo 4 – Resultados e Discussão: são apresentados os resultados de todos os testes efetuados para 19 instâncias de diferentes complexidades, incluindo vários parâmetros como a proximidade ao *makespan* ótimo, o tempo de processamento necessário para gerar a solução, a eficiência do teste e o número de tentativas necessárias. Seguidamente é efetuada uma análise estatística com objetivo de comparar os LLMs selecionados e compreender melhor os pontos fortes e os pontos fracos de cada um.
- Capítulo 5 – Conclusão: no final é feito um resumo de toda a informação abordada neste trabalho e das limitações presentes, procurando retirar conclusões relevantes para projetos futuros.

Introdução

2. Revisão Bibliográfica

Neste capítulo é apresentada a Revisão Bibliográfica, uma análise dos fundamentos técnicos que servem de base para esta dissertação. Primeiro é abordado o Escalonamento das Operações, identificando a nomenclatura tipicamente utilizada, tipos de problemas de Escalonamento e as formas de representação destes. De seguida, uma introdução aos conceitos gerais de Inteligência Artificial, com destaque em Large Language Models (LLMs) como ChatGPT, Copilot, Gemini e DeepSeek. No final do capítulo é realizada uma análise a casos de estudo que unem estes dois temas principais.

2.1. Escalonamento das Operações

O Escalonamento das Operações é definido como o processo de alocação das atividades necessárias para fabricar um produto, aos recursos disponíveis, planeando a sequência temporal das operações. O Escalonamento faz parte do sistema de Planeamento, Programação e Controlo da Produção (PPCP), especificamente da secção da Programação da Produção, atuando num período de tempo muito curto, mas fornecendo informação com grande nível de detalhe [1].

Demeulemeester et al. [2] define Escalonamento como a atribuição de uma ordem temporal em que as atividades do projeto devem ser processadas. Brucker [3] adiciona que a escolha desta programação deve ainda atender às limitações do problema.

Através de um cuidadoso Escalonamento das operações é possível reduzir os tempos da produção e obter maior estabilidade nos processos, evitando atrasos. Assim, esta etapa do PPCP é crucial na gestão industrial, especialmente em ambientes de produção complexos que contém um elevado número de atividades a competir pelos mesmos recursos.

O desafio de planear a execução de um conjunto de tarefas utilizando um número limitado de recursos é algo comum em quase todos os tipos de atividades numa organização, não apenas referindo ao contexto de produção. Assim, o Escalonamento é uma área estudada em detalhe há várias décadas, em diferentes setores industriais como o da petrolífera e gás, alimentar, transporte, farmacêutica, militar, etc. [4], [5].

2.2. Definição

Apesar de atualmente o escalonamento das operações poder ser aplicado em diversas áreas, nomeadamente em recursos computacionais, a origem do escalonamento surgiu na aplicação em ambientes industriais. Foi necessário definir inicialmente alguns termos e variáveis para melhor compreensão e comunicação deste assunto, que serão usados ao longo deste trabalho.

- Operação - Parte de uma tarefa que utiliza apenas um recurso.
- Tarefa/Atividade – Trabalho, que pode ser dividido em múltiplas operações, definido pela data de lançamento e data de entrega.
- Recurso – Máquina que executa operações.
- Oficina – Local de trabalho, onde se encontram os recursos disponíveis para realização das operações.

Um problema de escalonamento é definido por n tarefas $J = \{J1, J2, \dots, Jn\}$, divididas em operações $T = \{T1, T2, \dots, Tnm\}$, m recursos $P = \{P1, P2, \dots, Pm\}$ e s recursos adicionais $R = \{R1, R2, \dots, Rs\}$. Habitualmente, num determinado instante cada operação só pode ser processada numa máquina e cada máquina só pode estar a processar uma operação [6].

Para completar a definição do problema de escalonamento, são introduzidas as seguintes variáveis.

- Tempo de processamento (p_{ij}) – Tempo que demora à máquina i a processar a tarefa j .
- Data de lançamento (r_j) – Data em que é possível iniciar a tarefa j .
- Data de entrega (d_j) – Data-limite para completar a tarefa j .
- Peso (w_j) – Importância relativa da tarefa j .
- Data de conclusão (C_j) – Tempo que demora a tarefa j a sair do sistema após processamento na última máquina.
- Atraso (L_j) – Positivo quando a tarefa j é concluída depois da data-limite de entrega definida (atraso efetivo), e negativo quando a tarefa é concluída antes (antecipação).

$$L_j = C_j - d_j \quad (1)$$

- Atraso Positivo (T_j) – Atraso efetivo. Este valor nunca é negativo.

$$T_j = \max(L_j; 0) \quad (2)$$

- Atraso Negativo (E_j) – Antecipação. Este valor nunca é negativo

$$E_j = \max(d_j - C_j; 0) \quad (3)$$

- Penalidade Unitária (U_j) – Variável binária. Indica se a tarefa tem atraso positivo.

2.3. Fases do Escalonamento

Existem três secções num problema de Escalonamento de operações que, geralmente, devem ser analisadas na seguinte ordem: primeiro os problemas de Afetação/Alocação, que se referem à decisão de atribuir as operações aos recursos disponíveis, depois os problemas de Sequenciação, para definir a ordem das operações e por último os problemas de Calendarização quando os recursos e a ordem das operações estiverem determinados.

2.3.1. Problema de Afetação

No problema de afetação ou alocação, o objetivo é atribuir as operações aos recursos disponíveis, de forma a otimizar o uso dos recursos, tendo em conta a capacidade, prioridades e custos. O problema consiste em determinar quais operações vão ser efetuadas em que máquina, em cada fase do processo [6]. Estas máquinas podem não ser idênticas, ou, possuir características diferentes umas das outras, o que pode dificultar a afetação, favorecendo a realização de uma determinada operação numa máquina em vez de outra. Na Figura 1 está representado um exemplo de um problema de afetação.

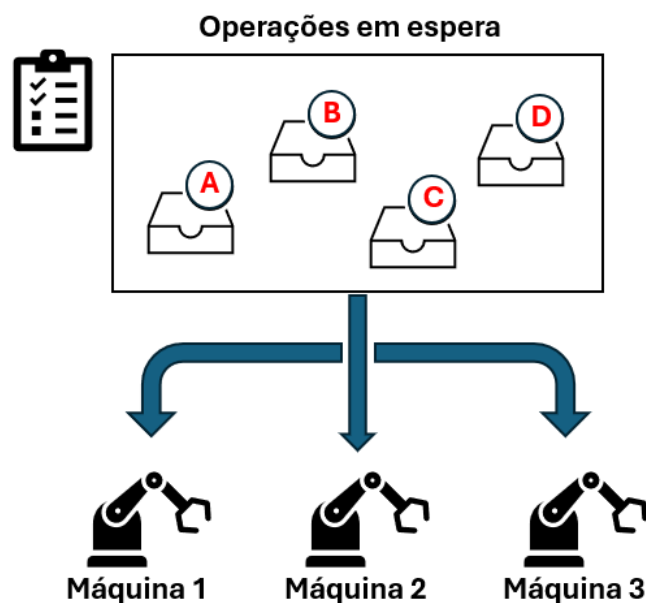


Figura 1 - Representação do Problema de Afetação (adaptado de [7])

Jacek Błażewicz et al. [6] descrevem o problema de afetação como “tendo n operações em $T = \{T_1, T_2, \dots, T_n\}$, m máquinas em $P = \{P_1, P_2, \dots, P_m\}$ e s recursos adicionais em $R = \{R_1, R_2, \dots, R_s\}$, é necessário atribuir as máquinas P e os recursos R às operações T com o objetivo de otimizar um indicador de desempenho”.

2.3.2. Problema da Sequenciação

Após a alocação das operações, o problema de sequenciação tenta encontrar a ordem de realização dessas operações em cada máquina de maneira a otimizar uma ou mais medidas de desempenho. Kasana & Kumar [8] definem o problema de sequenciação como “em que ordem um número finito de operações devem ser realizadas num número finito de máquinas, de forma a que o tempo total de conclusão das operações seja minimizado?”. Na Figura 2 está representado um exemplo de um problema de sequenciação numa máquina.

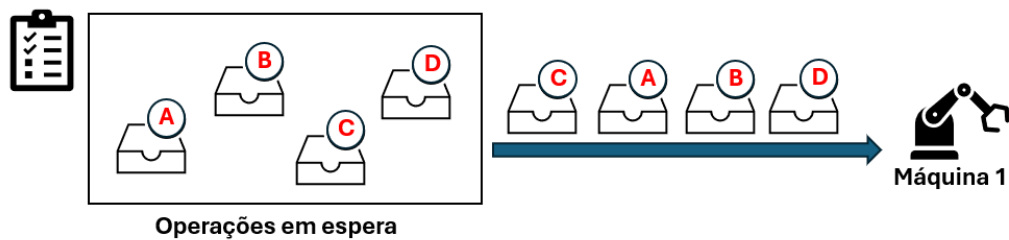


Figura 2 - Representação do Problema de Sequenciação (adaptado de [7])

O problema de sequenciação pode ser definido como “tendo n operações em $T = \{T1, T2, \dots, Tn\}$ e uma sequência de fabrico $S = \{S1, S2, \dots, Sn\}$, é necessário ordenar as operações T na sequência S para otimizar um indicador de desempenho”.

2.3.3. Problema de Calendarização

Com a sequência das operações determinada, é possível passar para a última fase do escalonamento das operações. O problema de calendarização usa a informação obtida pelos problemas de afetação e de sequenciação para distribuir as operações por um intervalo de tempo, e é um passo crucial no escalonamento para assegurar que não há sobreposições de operações na mesma máquina ao mesmo tempo. Este problema foca-se em planear o momento certo para iniciar o trabalho e, se possível, o momento de conclusão [9]. Na Figura 3 está representado um exemplo de um problema de calendarização.

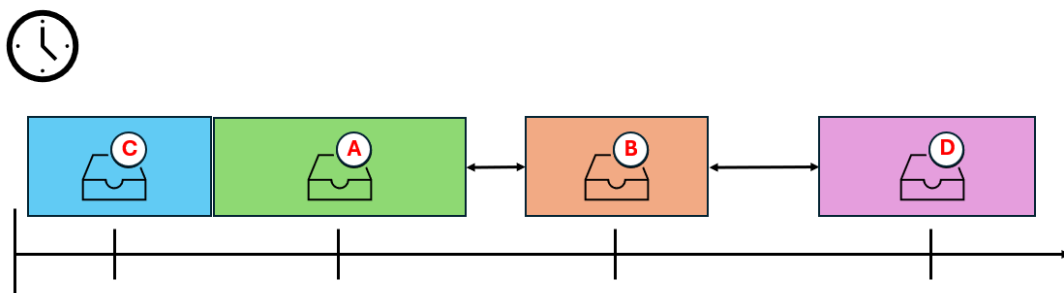


Figura 3 - Representação do Problema de Calendarização

O problema de calendarização pode ser definido como “tendo n operações em $T = \{T1, T2, \dots, Tn\}$, m máquinas em $P = \{P1, P2, \dots, Pm\}$, uma distribuição das operações de T em P , uma sequência de fabrico $S = \{S1, S2, \dots, Sn\}$ para cada máquina de P e um intervalo de tempo $[a,b]$, é necessário distribuir as operações de T pelo intervalo de tempo $[a,b]$ para otimizar um indicador de desempenho”.

2.4. Representação

Os problemas de Escalonamento em ambientes industriais baseiam-se sempre nos mesmos princípios, apesar disto eles podem apresentar propriedades e níveis de complexidade muito diferentes dependendo das características da oficina de trabalho, número de recursos disponíveis e forma como estão dispostos, e dos critérios de otimização do processo.

A representação de Conway [10] descreve os problemas através da notação dos quatro campos $A/B/C/D$. A representa o número de operações, B o número de recursos, C o ambiente de produção e D os critérios de otimização.

French [11] apresenta uma notação semelhante, com os campos $n/m/A/B$. Aqui n representa o número de operações, m o número de recursos, A o ambiente de produção e B os critérios de otimização.

A notação mais utilizada é a de Graham [12], com os campos $\alpha/\beta/\gamma$. Nesta classificação, α representa o ambiente de produção ou tipo de problema e contém um único valor. β descreve as características e restrições do processo, podendo conter vários valores, um único valor ou até ser nulo, indicando neste caso que todas as operações estão disponíveis para processamento. γ representa o critério de otimização, e tipicamente contém só um valor.

Já Pinedo [13] usa uma notação muito semelhante à de Graham, considerando os seguintes ambientes de produção em α : Máquina única, Máquinas paralelas, Flow-shop, Job-shop ou Open-shop. Estes tipos de problemas são abordados com maior detalhe no próximo ponto.

Segundo Pinedo, as restrições possíveis em β são:

- **Data de lançamento (r_j)** – Tarefa j apenas pode começar processamento depois da data r_j . Se este campo não existir, o processamento da tarefa pode começar a qualquer momento.
- **Interrupções (prmp)** – Indica que é possível parar o processamento de uma operação e iniciar o processamento de outra na mesma máquina, sem perder o progresso da primeira operação.
- **Precedências (prec)** – Requer que uma ou mais operações estejam completas antes de iniciar o processamento de outra.
- **Tempos de setup dependentes de sequência (s_{jk})** – Indica o tempo de preparação entre o processamento das operações j e k .
- **Famílias de operações (fmls)** – n operações pertencem a F famílias. As operações da mesma família podem ser processadas sem tempo de preparação entre elas.

- **Processamento de lotes (batch(b))** – Indica que uma máquina pode processar b operações simultaneamente.
- **Avárias de máquinas (brkdw)** – Implica que uma máquina pode avariar por períodos de tempo fixos.
- **Qualificação das máquinas (M_j)** – Representa o conjunto das máquinas que têm a capacidade de processar a operação j .
- **Permutações (prmu)** – Implica que a sequência de processamento das operações é fixa durante o sistema inteiro.
- **Sem espera (nwt)** – Indica que não pode existir tempo de espera entre operações.
- **Recirculação (rcrc)** – Indica que uma operação pode passar pela mesma máquina várias vezes.

Completando a notação de Pinedo, são apresentados os seguintes critérios de otimização:

- **Minimizar o Makespan ($\min C_{max}$)** – Reduzir o tempo de conclusão da última tarefa na última máquina.
- **Minimizar o Atraso máximo ($\min L_{max}$)** – Reduzir o maior atraso possível.
- **Minimizar o Tempo de conclusão ponderado total ($\min \sum w_j C_j$)** – Reduzir a soma dos tempos de conclusão pesados de n tarefas.
- **Minimizar o Atraso positivo ponderado total ($\min \sum w_j T_j$)** – Reduzir a soma dos atrasos positivos pesados de n tarefas.
- **Minimizar o Número de tarefas em atraso ponderado ($\min \sum w_j U_j$)** – Reduzir número total de tarefas com atraso positivo.

2.5. Ambiente de produção

2.5.1. Máquina Única

Representado por “1” na notação de Pinedo [13], o problema em máquina única é o mais simples possível, estando incluído por vezes nos problemas mais complexos [13]. Ou seja, o processo de resolução de alguns problemas em ambientes de produção mais complexos passa por resolver um problema de máquina única. Este problema apresenta tarefas com apenas uma operação a ser realizada na única máquina disponível, sendo apenas necessário definir a ordem de realização das tarefas através da sequenciação [14]. Deste modo, é irrelevante falar na alocação para problemas em máquina única.

Segundo Baker & Trietsch [9] o problema básico de máquina única apresenta as seguintes características:

1. Existem n tarefas com uma só operação disponíveis simultaneamente para processamento quando $t = 0$.
2. Máquina pode processar no máximo uma tarefa de cada vez.
3. Tempos de *setup* das tarefas são independentes da sequência das tarefas e estão incluídos no tempo de processamento.
4. As características das tarefas são determinísticas e conhecidas.
5. Máquina está sempre disponível (sem avarias).
6. Máquina não fica parada à espera de tarefas.
7. Assim que uma operação começa, procede sem interrupções.

2.5.2. Máquinas Paralelas

O problema em máquinas paralelas apresenta várias tarefas com uma só operação, que precisam, inicialmente, de ser alocadas às máquinas disponíveis. Depois deste passo, o problema transforma-se em vários problemas em máquina única, abordados anteriormente.

Pinedo [13] faz a distinção entre máquinas paralelas idênticas “Pm”, onde as tarefas podem ser processadas em qualquer máquina, máquinas paralelas com velocidades diferentes “Qm”, onde cada máquina tem uma velocidade predefinida, ou máquinas paralelas diferentes “Rm”, onde a velocidade depende de qual tarefa é processada em que máquina.

2.5.3. Flow Shop

Com a notação em α “Fm”, Flow Shops apresentam m máquinas em série, sendo que cada tarefa é dividida em várias operações, cada uma devendo ser processada numa máquina. As tarefas devem passar por todas as máquinas do sistema numa determinada ordem igual para todas as tarefas, tipicamente seguindo a disciplina de First In First Out (FIFO) onde uma tarefa não pode “ultrapassar” outra na fila de espera. Assim, numa Flow Shop, cada operação tem exatamente um antecessor e um sucessor [9], [13].

Uma Flow Shop com FIFO deve ter um limite de tarefas a ser executadas ao mesmo tempo e, quando esse limite for atingido, a entrada de novas tarefas deve ser interrompida. Isto assegura o objetivo do fluxo contínuo [15].

Existe ainda uma generalização do ambiente de Flow Shop e de Máquinas Paralelas, designado Flow Shop Flexível “FFc”, no qual em vez de ter m máquinas em série tem-se c etapas em série, cada etapa com m máquinas em paralelo. Neste ambiente de produção existe igualmente uma ordem que as tarefas devem seguir, no entanto essa ordem apenas aplica-se às etapas. Em cada etapa a tarefa pode ser processada por qualquer uma das máquinas disponíveis [13].

2.5.4. Job Shop

Uma Job Shop “Jm” é caracterizada por m máquinas e n tarefas compostas por várias operações, onde cada tarefa segue um percurso específico, pré-determinado. Neste ambiente de produção as tarefas podem ser processadas mais do que uma vez em cada máquina, existindo a possibilidade de recirculação no campo β . Assim, distingue-se de uma Flow Shop, por não apresentar uma máquina de início e uma máquina terminal idêntica para todas as tarefas [9], [13].

Como no caso anterior, existe uma generalização do ambiente Job Shop e de Máquinas Paralelas (Job Shop Flexível “FJc”) que não tem um único conjunto m de máquinas, mas sim várias etapas c com m máquinas idênticas paralelas.

2.5.5. Open Shop

Situação semelhante a uma Job Shop, no entanto as operações não apresentam uma ordem específica de processamento. Não existem restrições do percurso das tarefas pela oficina, sendo assim um ambiente de produção mais flexível do que uma Flow Shop ou Job Shop [16].

2.6. Natureza do Escalonamento

Originalmente os problemas de Escalonamento eram abordados pela academia de uma forma teórica, como modelos estáticos. Nestes modelos todas as operações são conhecidas no início do planeamento e a sequência de processamento mantém-se fixa durante o processo de fabricação. No entanto, estes modelos não são aplicáveis na maioria dos casos de produção real e uma quantidade crescente de autores já abordam os problemas com o seu dinamismo.

Em 1996, Shukla & Frank Chen [17] realizaram um inquérito sobre o controlo inteligente em tempo real num sistema de produção flexível, mostrando que, na altura, a teoria e a prática de Escalonamento mostravam pouca correspondência entre si. Isto porque, na prática, existem restrições que transformam o problema de Escalonamento estático em Escalonamento dinâmico. Suresh & Chaudhuri [18] dividem estas restrições em cinco tipos: políticas e objetivos organizacionais, limitações físicas, restrições causais, restrições de disponibilidade e restrições de preferência. Destas, as restrições físicas, como tempos de setup, e as restrições de disponibilidade, como tempo de máquina parada, número de turnos, recursos necessários e inventário, são as que podem originar perturbações no planeamento.

Já Ouelhadj & Petrovic [19] definiram um ambiente dinâmico como aquele onde eventos imprevisíveis e inevitáveis, como falhas nas máquinas, introdução de trabalhos mais urgentes ou alterações nas datas, ocorrem em tempo real e podem causar uma mudança no planeamento. Pensa-se que numa cadeia de abastecimento internacional de larga escala podem existir cerca de 1 milhão de perturbações e alterações não planeadas por ano, e os gestores ocupam 40% a 60% do tempo a resolver estas perturbações [20], [21].

Já no que diz respeito a incerteza no Escalonamento, é ainda possível diferenciar entre Escalonamento Determinístico e Escalonamento Estocástico. O primeiro assume que os

parâmetros do problema são conhecidos antecipadamente com precisão. É comum em contextos acadêmicos e em problemas bem definidos, permitindo maior controle e otimização dos processos. No entanto, pode não refletir a incerteza inerente aos ambientes produtivos verdadeiros. Já o Escalonamento Estocástico considera que esses parâmetros são incertos e apenas conhecidos no momento em que ocorrem. Para lidar com essa imprevisibilidade, utilizam-se distribuições estatísticas que modelam a incerteza. Esta abordagem aproxima-se mais da realidade industrial, sendo mais flexível, mas também mais complexa de implementar [13].

2.7. Inteligência Artificial

Um dos temas de maior relevância da atualidade é a Inteligência Artificial (IA), um campo interdisciplinar estudado e discutido em diversas áreas como na ciência da computação, engenharia, medicina, economia, e ciências sociais. Com a crescente aplicação e impacto transformador da IA, este tema tem sido objeto de análise diária, tanto no avanço de tecnologias como na compreensão dos efeitos éticos, sociais e econômicos.

As primeiras referências sobre IA remontam a mais de 60 anos atrás. John McCarthy, um dos pioneiros de IA, foi o primeiro a definir o termo “Inteligência Artificial”, em 1955, como: *“The goal of AI is to develop machines that behave as though they were intelligent”*. Esta definição foi, no entanto, vista como insuficiente e, mais tarde, em 1993, Elaine Rich [22] sugeriu a seguinte definição: *“Artificial Intelligence is the study of how to make computers do things at which, at the moment, people are better”*.

Talvez uma definição mais moderna seria a aplicação de algoritmos que proporcionam às máquinas a habilidade de resolver problemas que tradicionalmente necessitavam de inteligência humana [23].

Nos últimos anos o enorme volume de dados, em vários formatos, a ser gerado a uma velocidade cada vez maior, exigiu um desenvolvimento acelerado de novas tecnologias e o aumento da capacidade de processamento computacional [24], [25]. Com este impulsionado desenvolvimento tecnológico, IA teve recentemente um crescimento exponencial, tendo passado de especulação para aplicação em vários setores, como em veículos autônomos, recomendações de aplicações de streaming, anúncios e detecção de fraude [26], [27].

Brynjolfsson & McAfee [28] concluem que os maiores avanços na IA têm sido na área da percepção, com o reconhecimento de voz (ex. Siri, Alexa) e reconhecimento de imagens (ex. Google Lens), e na área do raciocínio e resolução de problemas. Agrawal, Gans & Goldfarb [29] sugerem que o maior impacto de IA é a previsão num mundo incerto, concedendo um melhor poder de decisão. A Figura 4 apresenta alguns dos desenvolvimentos mais importantes na área da IA do último século.

Evolução da Inteligência Artificial

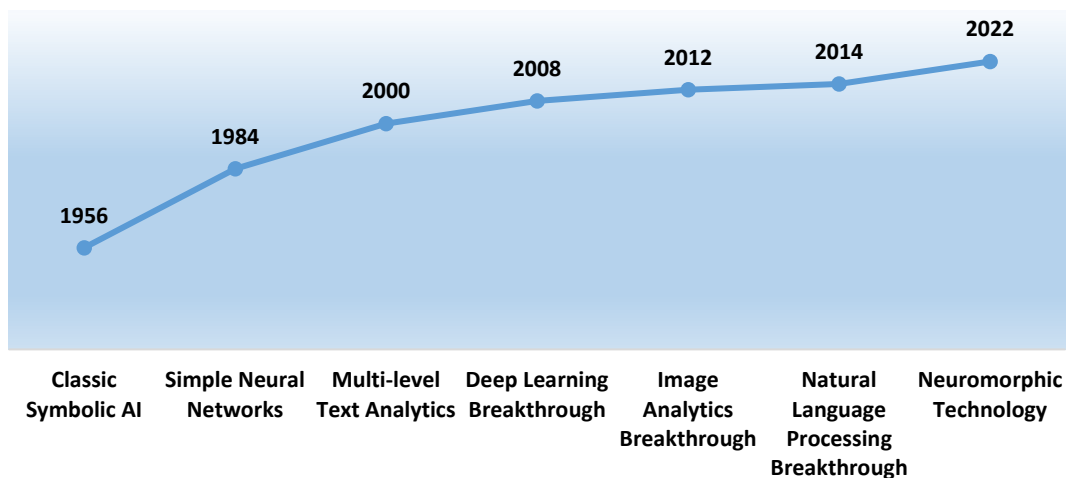


Figura 4 - Evolução da Inteligência Artificial (adaptado de [30])

Com esta rápida evolução da IA, muitas empresas começaram a aplicar a tecnologia nos seus processos nos últimos anos. Em 2022 foi estimado que 34% de todas as tarefas realizadas nas empresas eram desempenhadas de forma automática por máquinas, com previsão de crescimento para 42% em 2027 [31].

2.8. Machine Learning

Machine Learning (ML) é uma sub-divisão da Inteligência Artificial que consiste na criação de um conjunto de algoritmos computacionais com a capacidade de aprender através do ambiente envolvente, e de processar e extrair informações como tendências e relações históricas dos dados [32], [33], [34].

Dependendo da forma como os modelos aprendem e extraem informação, o tipo de dados disponíveis e o objetivo do modelo, existem diferentes tipos de cenários de aplicação de ML. Estas podem ser categorizadas como Supervised Learning (SL), Unsupervised Learning (UL) ou Reinforcement Learning (RL) [35], [36].

- **Supervised Learning** – SL é um método onde o modelo recebe um conjunto de informação rotulada como dados de treino e aprende a realizar previsões. Este cenário é geralmente utilizado em problemas de classificação ou regressão, devido à capacidade de mapear entradas para saídas com base em algoritmos específicos. Técnicas comuns incluem regressão linear e árvores de decisão. A informação de entrada pode ser um vetor, um número, uma imagem, um email, uma série temporal ou um gráfico. A informação de saída deve ser uma variável categórica ou nominal – classificação, ou um valor real – regressão [35], [36], [37].
- **Unsupervised Learning** – Com UL o modelo recebe exclusivamente dados não rotulados e procura identificar padrões implícitos nos dados de entrada. Este método é frequentemente utilizado em problemas como *clustering*, que agrupa elementos com

características semelhantes, e redução de dimensionalidade, que simplifica dados mantendo as informações essenciais. Técnicas comuns incluem *K-Means clustering* e *Principal Component Analysis*. Uma vez que não existem exemplos rotulados neste contexto, pode ser difícil avaliar quantitativamente o desempenho do modelo. Este tipo de abordagem é particularmente útil para explorar e organizar grandes conjuntos de dados de forma automática, sem depender de supervisão humana [35], [36].

- **Reinforcement Learning** – RL é um método onde um agente aprende a tomar decisões ao interagir com um ambiente, recebendo recompensas ou penalizações imediatas com base nas suas ações. O objetivo do agente é maximizar a recompensa acumulada ao longo de várias interações, o que requer um equilíbrio entre a exploração de novas ações para adquirir mais informações e a exploração do conhecimento já obtido para obter recompensas. Apesar da ausência de feedback de longo prazo, este processo contínuo de interação com o ambiente torna a RL uma abordagem poderosa para resolver problemas dinâmicos e incertos [35], [36].

Estes tipos de aprendizagem podem ser usados individualmente ou combinados, dependendo das necessidades do problema e dos dados disponíveis. De forma a demonstrar a utilidade destes tipos, na Tabela 1 estão retratadas possíveis aplicações para cada tipo de problema.

Tabela 1 - Exemplos de aplicações dos diferentes tipos de aprendizagem

Supervised Learning	
Classificação	Regressão
Deteção de spam em e-mails, no qual mensagens são rotuladas como "spam" ou "não spam", fazendo o modelo aprender a diferenciar categorias a partir de dados fornecidos.	Previsão do preço de uma casa com base no tamanho e na localização, formando uma regressão linear.
Unsupervised Learning	
Clustering	Redução de dimensionalidade
Agrupamento de clientes, por parte das empresas, com base em comportamentos, preferências e padrões de compra.	Identificação de associações genéticas com doenças, otimizando a análise de dados de sequenciamento genético.
Reinforcement Learning	
Robótica	Jogos
Ensino de máquinas a realizar tarefas complexas como navegação e manipulação de objetos.	Ensino de agentes a aplicar estratégias otimizadas para vencer os oponentes.

Um dos primeiros estudos na área de Machine Learning pode ser atribuído a Warren McCulloch e Walter Pitts, em 1943 [38], quando propuseram o primeiro modelo matemático de uma rede neuronal, destacando o seu potencial para simular processos lógicos de pensamento. Este modelo foi fundamental para a conceção de redes neuronais artificiais, fornecendo uma base teórica para a computação inspirada na atividade cerebral e apoiando os futuros avanços no campo de Machine Learning. Alguns anos depois, em 1950, Alan Turing [39] questionou: “Podem as máquinas pensar?”. Através do Teste de Turing, foi proposto um método experimental para avaliar se o comportamento de uma máquina poderia ser idêntico ao comportamento humano. Este conceito gerou vários debates filosóficos e inspirou investigações sobre algoritmos de aprendizagem.

Nas décadas de 1940 e 1950 surgiram outros avanços importantes para ML, nomeadamente a aprendizagem Hebbiana, apresentada por Donald Hebb em 1949 [40], que introduziu a ideia de que as conexões sinápticas se fortalecem com a ativação repetida, o *Perceptron*, criado por Frank Rosenblatt em 1958 [41], que tratava-se de um sistema capaz de classificar dados de entrada com base em pesos ajustáveis, e finalmente o trabalho pioneiro de Arthur Lee Samuel sobre o seu jogo de damas e aprendizagem baseada em experiência, em 1959 [42], onde introduziu pela primeira vez o termo Machine Learning.

Nos últimos anos, ML evoluiu rapidamente, tornando-se um pilar no desenvolvimento tecnológico. Os avanços desta ferramenta proporcionaram a criação de GANs (Generative Adversarial Networks), introduzidas em 2014 por Ian Goodfellow et al. [43]. Estes modelos envolvem dois componentes: um gerador que cria dados sintéticos e um discriminador que os avalia em relação aos dados reais. Através de GANs é possível criar imagens e vídeos altamente realistas. Outro desenvolvimento fundamental para IA foi feito em 2017 com o artigo "*Attention Is All You Need*", de Vaswani et al. [44], que introduziu o modelo *Transformer*, uma arquitetura que utiliza mecanismos de atenção para processar sequências de dados. Este modelo tornou-se a base para sistemas como o BERT (Bidirectional Encoder Representations from Transformers) e GPT (Generative Pretrained Transformer), revolucionando tarefas de tradução, sumarização e geração de texto.

Hoje, sistemas inteligentes que oferecem capacidades de inteligência artificial frequentemente dependem de Machine Learning [45]. A integração de ML em diversas áreas não só melhorou a eficiência e precisão dos sistemas, mas também expande as capacidades da inteligência artificial, permitindo soluções inovadoras para problemas complexos e impulsionando o progresso tecnológico em múltiplos setores. É possível encontrar aplicações desta tecnologia no campo da engenharia aeroespacial [46], visão computacional [47], reconhecimento de padrões [48], aplicações biomédicas e médicas [49], [50], biologia computacional [51], ecologia [52], finanças [53] e entretenimento [54].

2.8.1. Deep Learning

Com os constantes desenvolvimentos na área de Machine Learning nos últimos anos, as expectativas que a população tem das máquinas têm aumentado cada vez mais. Deep Learning (DL) é uma tentativa de responder a estas expectativas [32]. O DL é uma secção de ML que usa Redes Neurais Artificiais com várias camadas, conseguindo atingir um desempenho superior ao modelar padrões complexos [45], [36]. Na Figura 5 observa-se a relação entre Inteligência Artificial, Machine Learning e Deep Learning no que se refere ao nível de complexidade dos algoritmos de processamento dos dados.

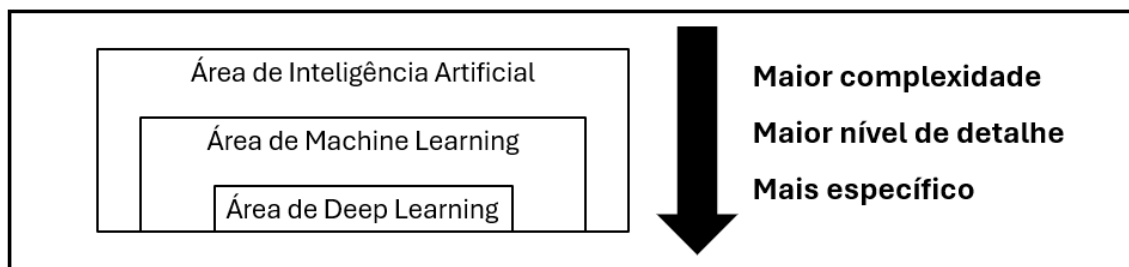


Figura 5 - Comparação de Deep Learning com Inteligência Artificial e Machine Learning (Adaptado de [32])

Geoffrey Hinton et al. [55] introduziram, em 2006, o conceito de Deep Learning com a utilização de *Deep Belief Networks* (DBNs), demonstrando como camadas profundas de Redes Neurais podem ser treinadas eficientemente utilizando um pré treino com Unsupervised Learning seguido de aperfeiçoamento com Supervised Learning. Este método reduziu drasticamente problemas de sobreajuste, permitindo que redes mais complexas aprendessem com grandes volumes de dados.

Nos anos seguintes, verificou-se uma rápida evolução deste campo, em grande parte impulsionada pelo aumento do poder computacional e pela disponibilidade de grandes bases de dados. Arquiteturas introduzidas no passado começaram a ser aperfeiçoadas, como as *Convolutional Neural Networks* (CNNs) [56], que revolucionaram o processamento de imagens, e as *Recurrent Neural Networks* (RNNs) [57], que se destacaram em tarefas de séries temporais e linguagem natural.

Estas redes, mais concretamente chamadas de Redes Neurais Artificiais (ANNs), inspiradas pelo sistema nervoso humano (unidade computacional de um cérebro – neurónio), são compostas por nós interligados que processam dados através de múltiplas camadas [36], [58]. Os primeiros modelos possuíam redes de arquitetura simples, como o *Perceptron*, que apenas apresentava uma ligação entre duas camadas de nós, uma de entrada e uma de saída. Já os modelos mais recentes possuem camadas escondidas que ajudam a encontrar padrões mais complexos, tratando-se de *Deep Neural Networks* (DNNs). Estas redes são caracterizadas por uma “profundidade”, que determina o número de camadas escondidas que possuem, variando de duas a milhares [58], [59]. Na Figura 6 é apresentado um exemplo da estrutura de uma Rede Neuronal com duas camadas escondidas que, apesar de ser considerada DNN, é ainda relativamente simples.

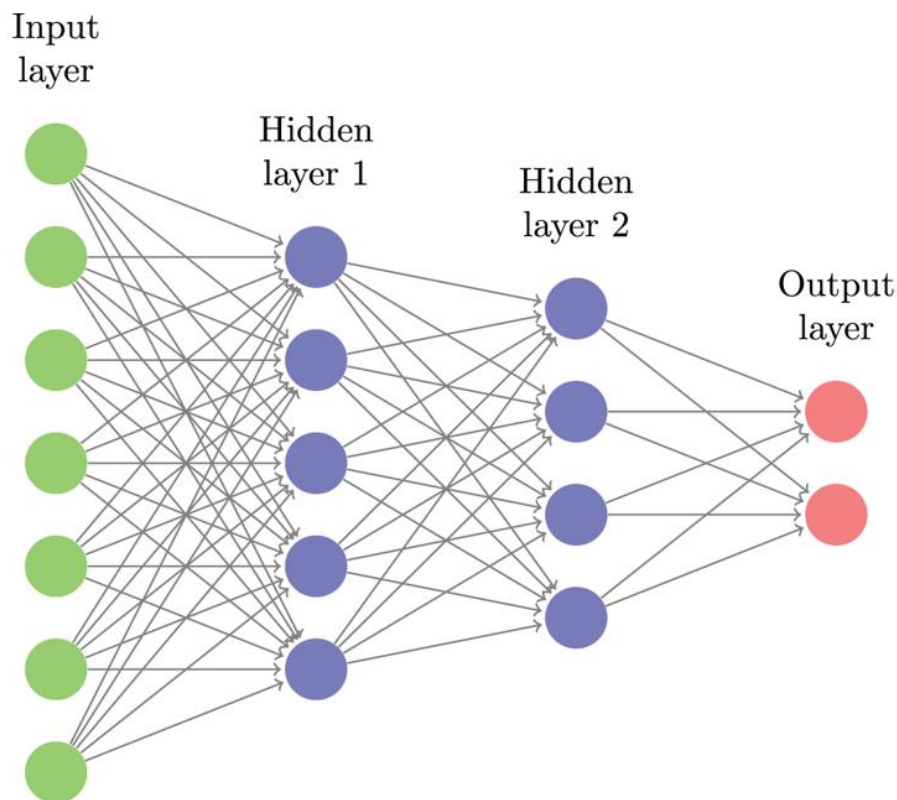


Figura 6 - Exemplo da estrutura de uma Deep Neural Network (DNN) [59]

DL tem um impacto significativo em diversas aplicações, destacando-se na visão computacional, onde possibilitou avanços notáveis em classificação de imagens e detecção de objetos, e no processamento de linguagem natural, melhorando tarefas como tradução e análise de sentimento. Tem também alcançado progressos importantes em áreas como reconhecimento de fala, reconhecimento de texto, leitura labial, diagnóstico assistido por computador, reconhecimento facial, entre outras [36], [60].

2.9. Large Language Models

Como já foi referido, um dos principais usos de Deep Learning é no Processamento de Linguagem Natural (PLN), um subcampo da IA dedicado a capacitar as máquinas a compreender, processar e gerar linguagem humana, que abrange diversas técnicas, incluindo abordagens mais tradicionais como análise gramatical e de semântica ou modelagem de tópicos.

Durante muitos anos, as *Convolutional Neural Networks* (CNN) e as *Recurrent Neural Networks* (RNN) apresentavam um desempenho superior a qualquer tipo de arquitetura de Rede Neuronal. No entanto, a rápida evolução das redes e de DL ajudaram à melhoria da análise e geração de texto, tornando possível capturar nuances semânticas e contextuais da linguagem natural, e, com a introdução da arquitetura *Transformer* por Vaswani et al. em 2017 [44], este campo foi revolucionado.

Os Modelos de Linguagem de Grande Escala (LLMs) são implementações avançadas no PLN que utilizam a arquitetura *Transformer*. Estes modelos surgiram como sistemas de IA altamente avançados e treinados em vastas quantidades de dados, permitindo-lhes gerar texto coerente e relevante [61]. LLMs demonstram uma versatilidade sem precedentes no PLN ao conseguir realizar uma grande quantidade de tarefas de forma eficiente, como responder a perguntas, tradução automática e até mesmo criar conteúdo criativo [36], [61].

Em menos de uma década, LLMs já se tornaram uma peça central não só das empresas, com aplicações em várias secções desde o atendimento ao cliente à análise de dados e à automação das tarefas, como também do dia-a-dia das pessoas. Em termos de investigação e desenvolvimento nesta área, é possível ver uma tendência ascendente de artigos publicados por ano na Figura 7.

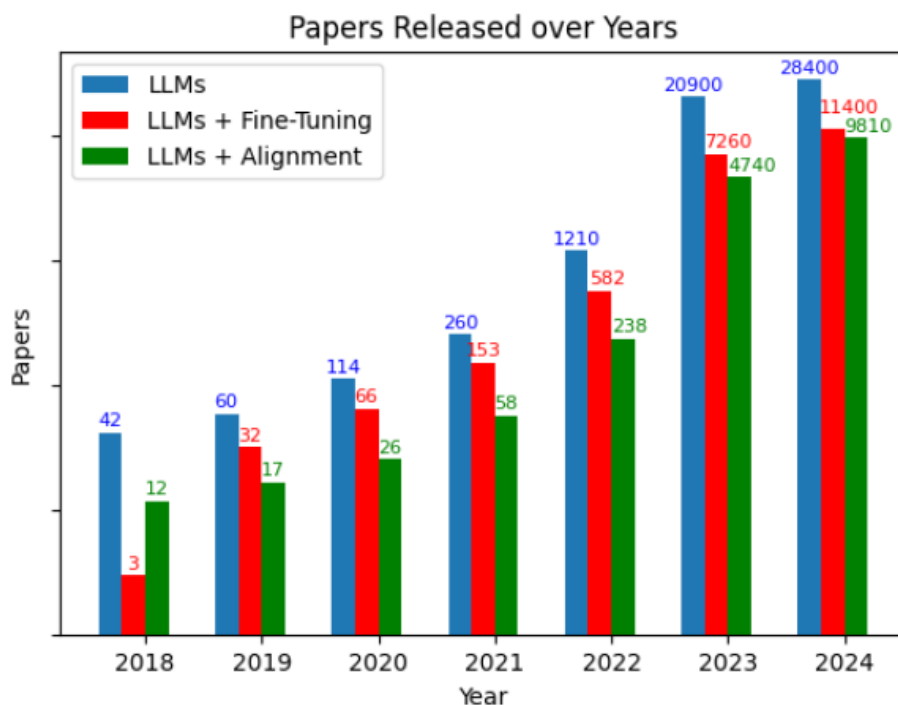


Figura 7 - Tendência de artigos publicados por ano sobre "Large Language Models" [61]

LLMs têm a habilidade de realizar várias tarefas complexas com um desempenho a nível humano ou superior, no entanto, isto só é possível com o uso de *hardware* ótimo, custos elevados, e um processo lento de treino, indispensável para a melhoria do modelo [61]. LLMs atuais passam inicialmente por uma fase de pré-treino com Self-Supervised Learning, onde tentam apenas prever qual a palavra seguinte com base no texto anterior e utilizando textos comuns da internet, considerados dados não rotulados. De seguida, é necessário o ajuste do modelo, numa fase com Supervised Learning, isto é, o humano introduz dados rotulados manualmente como perguntas e respostas pré-determinadas para ajudar a gerar respostas mais precisas e relevantes [62]. A este ponto, apesar do LLM já pode funcionar corretamente, é importante adicionar uma etapa com Reinforcement Learning, onde os utilizadores classificam as respostas quanto à sua qualidade, para alinhar o comportamento do modelo às expectativas humanas.

2.9.1. ChatGPT

O ChatGPT (*Chat Generative Pre-Trained Transformer*) é sem dúvida o LLM mais popular atualmente disponível ao público. Criado pela OpenAI, e lançado a 30 de novembro de 2022, este sistema consegue efetuar uma grande variedade de atividades de PLN como traduções, responder a perguntas e completar textos, tudo isto enquanto adapta-se ao humor da conversa. [63], [64].

ChatGPT tornou-se fenómeno cultural em menos de 2 meses após o lançamento [65], obtendo 1 milhão de utilizadores nos primeiros 5 dias. Este rápido crescimento está demonstrado na Figura 8, comparando-o com o de outras plataformas. Neste momento já tem mais de 100 milhões de utilizadores recorrentes, com 3 mil milhões de visitas no site por mês, tornando-a uma das maiores plataformas online [66].

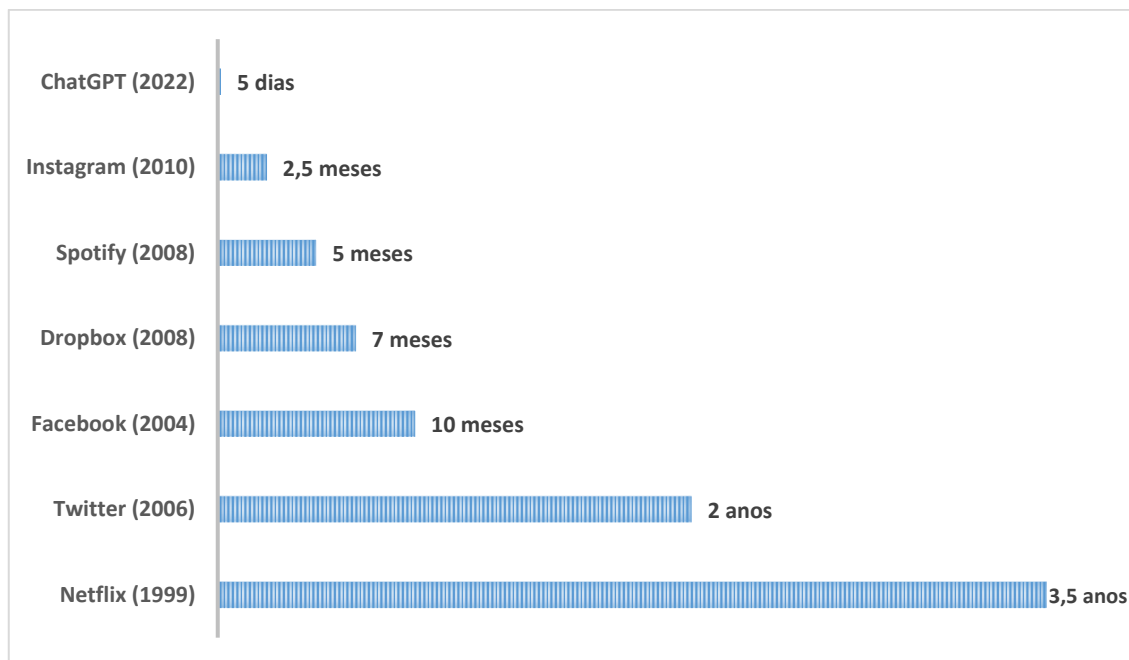


Figura 8 - Tempo que plataformas online demoraram a atingir 1 milhão de utilizadores (adaptado de [66])

Também no setor profissional é possível observar o enorme impacto que o ChatGPT tem demonstrado. Empresas de diferentes setores têm recorrido ao ChatGPT para automatizar tarefas administrativas, apoiar equipas de atendimento ao cliente, gerar relatórios técnicos e até como suporte à tomada de decisão. Um dos pontos mais valorizados é a sua capacidade de acelerar a análise de grandes quantidades de informação e de fornecer respostas rápidas, permitindo ganhos de eficiência operacional. A Figura 9 representa a percentagem de pessoas, em vários setores de trabalho, que utilizam, ou já utilizaram ChatGPT, e que conhecem a plataforma mas nunca a usaram. Enquanto que o uso varia um pouco entre profissão, é aparente que quase toda a gente já ouviu pelo menos falar da plataforma.

Adoption of ChatGPT across Occupations

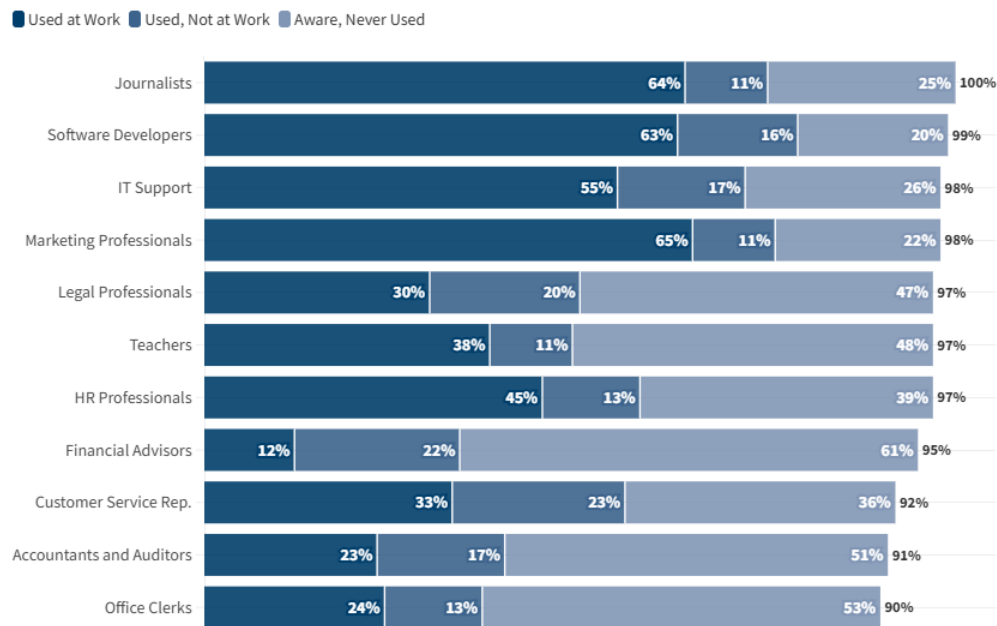


Figura 9 - Uso de ChatGPT por área profissional [67]

Com as maiores empresas de tecnologia a investirem nesta área na atualidade, a OpenAI destaca-se ainda pela criação do modelo de linguagem GPT-3, em 2020, que serve como base do ChatGPT [64]. No entanto o GPT-3 foi treinado inteiramente por Unsupervised Learning com uma grande quantidade de conversas, o que levava a regularmente apresentar respostas não esperadas [68]. Mais tarde o modelo evoluiu para o GPT-3.5, que já passa por uma fase de Supervised Learning para além de receber feedback humano, melhorando drasticamente o seu desempenho.

Atualmente o ChatGPT já disponibiliza o modelo GPT-4, um dos LLMs mais avançados, capaz de processar imagens e documentos de texto. Foi criada uma série de testes destinados a humanos, para avaliar as capacidades deste modelo. Na maioria dos casos o GPT-4 excedeu os resultados das pessoas que realizaram o teste, incluindo numa simulação de exame da ordem de advogados, onde obteve um resultado melhor que 90% dos humanos [69]. Na Figura 10 estão apresentados os resultados destes testes de vários temas académicos para GPT-3.5, GPT-4 e GPT-4 (sem capacidade de análise de imagens).

Apesar de todas estas vantagens do ChatGPT, é importante referir alguns aspetos a ter atenção e algumas limitações desta ferramenta. Com a capacidade de rapidamente gerar um manuscrito com informação pesquisada online, é possível que muitos autores estejam a escrever excertos “não originais” sem verificação cuidada, cometendo plágio e indo contra as regras editoriais [65]. É possível ainda que o próprio ChatGPT forneça informação errada, ou respostas fora das normas de segurança e privacidade. Para além disso, este tipo de modelos avançados são computacionalmente exigentes e requerem um hardware e uma infraestrutura poderosos para funcionar [63].

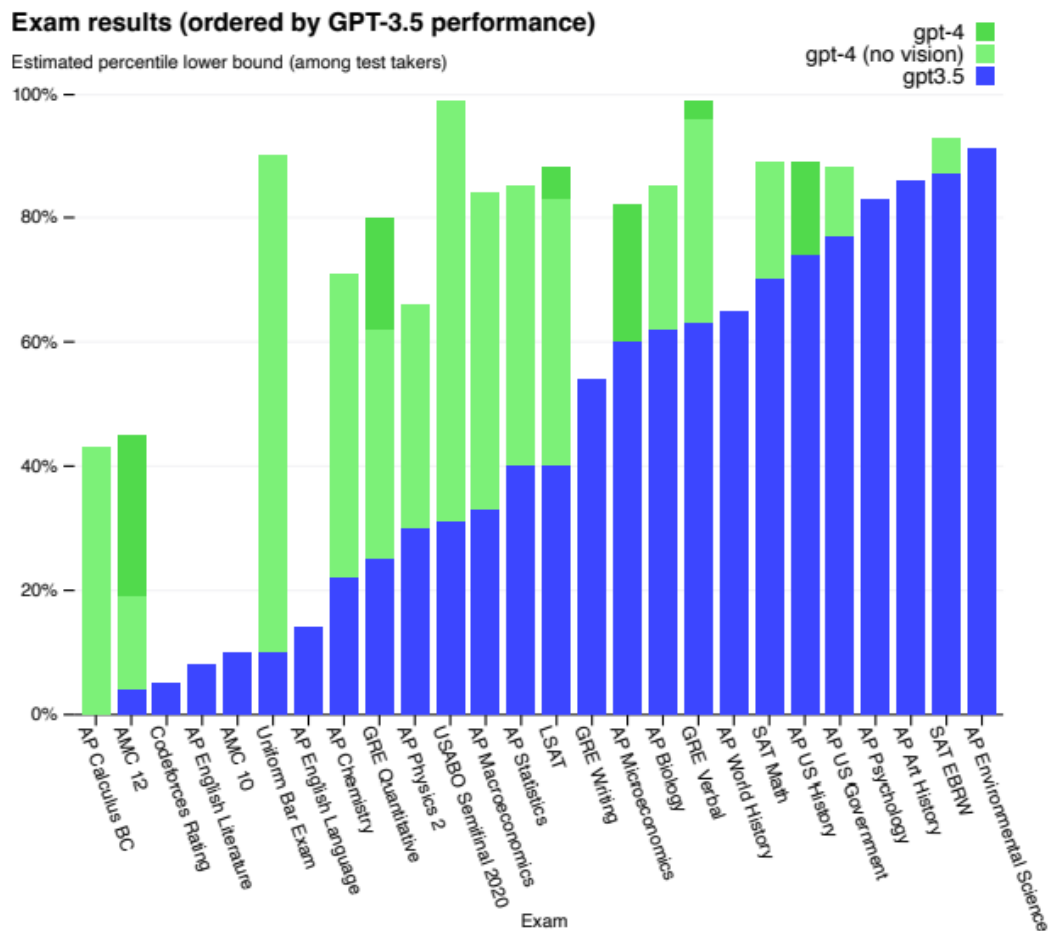


Figura 10 - Desempenho de GPT-3.5, GPT-4 e GPT-4 (sem visão) em vários testes acadêmicos

2.9.2. Copilot e Gemini

Nos últimos anos, a Microsoft também tem investido significativamente no desenvolvimento de soluções baseadas em IA, e o Copilot é um dos exemplos deste investimento. Criado em parceria com a OpenAI e lançado oficialmente a 7 de fevereiro de 2023, o Copilot é uma ferramenta de assistência baseada no modelo Microsoft Prometheus, que por sua vez utiliza a infraestrutura do GPT-4, aproveitando o conhecimento adquirido pelos LLMs da OpenAI para oferecer sugestões contextuais inteligentes aos utilizadores [70].

Comparativamente ao ChatGPT, o Copilot diferencia-se pela sua integração nativa em ferramentas empresariais e pela personalização do seu funcionamento conforme o contexto da aplicação. O Copilot apresenta implementação em diversas plataformas, como no GitHub, onde tem revolucionado a forma como programadores escrevem código e desenvolvem software, permitindo auto-completar funções, sugerir correções e melhorias no código e aumentar a eficiência na programação [71]. Já a sua implementação no Microsoft 365 permite auxiliar na redação de documentos, organização de e-mails e análise de dados no Excel, tornando as tarefas empresariais mais rápidas e intuitivas.

Desenvolvido pela Google DeepMind, o Gemini representa a mais recente aposta da empresa na evolução dos LLMs. Inicialmente chamado de Bard, e usando o modelo LaMDA, o *chatbot* foi lançado em março de 2023 com um desempenho limitado, e em dezembro do mesmo ano a empresa concebeu o LLM Gemini para competir diretamente com o GPT-4, trazendo inovações na sua arquitetura e capacidades multimodais. Este modelo recebeu destaque em especial pelo uso de *Reinforcement Learning* com *feedback* humano otimizado, levando a uma melhor precisão e confiabilidade das respostas, e pela multimodalidade nativa, ou seja, a capacidade de interpretar e apresentar informações em forma de texto, imagens, vídeos, áudio ou PDFs, como característica principal desde a sua criação [72].

No entanto o Gemini tem sido alvo de reações mistas pelo seu desempenho, e até gerou controvérsia em fevereiro de 2024 quando utilizadores nas redes sociais relataram que estava a gerar imagens historicamente imprecisas, apresentando incorretamente figuras históricas como pessoas racializadas [73].

Com o intuito de comparar o desempenho do Copilot, do Gemini e do ChatGPT entre si, Hochmair et al. Hochmair et al. [74] realizaram um estudo em janeiro de 2024 focado na exatidão das respostas destes LLMs a uma série de tarefas abrangendo temas como geografia, matemática e programação. A Tabela 2 apresenta os resultados deste estudo.

Tabela 2 - Exatidão dos LLMs em sete tarefas de diferentes categorias (adaptado de [74])

Categoria	ChatGPT-4	Gemini	Copilot
Literacia espacial	15/16 (93.8%)	11/16 (68.8%)	13/16 (81.3%)
Conceitos de SIG	10/10 (100%)	9/10 (90%)	10/10 (100%)
Mapeamento	6/10 (60%)	0/10 (0%)	2/10 (20%)
Interpretação de funções	10/10 (100%)	7/10 (70%)	8/10 (80%)
Explicação de código	10/10 (100%)	8/10 (80%)	10/10 (100%)
Programação	5/10 (50%)	3/10 (30%)	7/10 (70%)
Raciocínio espacial	2/10 (20%)	4/10 (40%)	4/10 (40%)
Total	58/76 (76.3%)	42/76 (53.3%)	54/76 (71.1%)

O autor concluiu que para estas tarefas, na data de realização do estudo, o ChatGPT-4 mostrou um desempenho superior comparado com outros LLMs, com uma maior taxa de respostas corretas. No entanto, teve o pior resultado em raciocínio espacial e cometeu erros graves em programação. O Copilot apresentou o segundo melhor desempenho, com especial destaque na capacidade de gerar código. A taxa de exatidão do Gemini foi inferior à dos outros dois, apresentando apenas um raciocínio espacial melhor que o ChatGPT-4.

2.9.3. DeepSeek

Recentemente outro modelo de IA surgiu, tornando-se rapidamente um dos principais tópicos de discussão nesta área. Desenvolvido pela empresa chinesa do mesmo nome, o DeepSeek foi lançado em janeiro de 2025 e imediatamente revolucionou o setor ao oferecer um desempenho comparável ao do ChatGPT, mas de forma gratuita.

O DeepSeek destaca-se relativamente a outros LLMs por ser *open-source*, permitindo uma adaptação sem precedentes por parte da comunidade, e pela integração do método *Chain-of-Thought* (CoT) para resolver problemas mais complexos, explicando detalhadamente o raciocínio necessário para chegar à resposta, passo a passo [75]. Apesar de ser possível levar outros LLMs a aplicarem este método às respostas, isso normalmente requer uma construção de *prompts* mais cuidadosa. A vantagem óbvia da utilização do CoT é a obtenção de respostas mais estruturadas que ajudam o utilizador a compreender o assunto, no entanto isto pode ser visto como uma desvantagem quando o objetivo do utilizador é obter respostas rápidas e concisas [76].

A versão atual, DeepSeek-R1, passou por várias fases de treino, desde *Unsupervised Learning* através de enormes bases de dados e *Supervised Fine-Tuning*, até ao *Reinforcement Learning* com *feedback* humano. Este modelo diferencia-se pela aplicação da técnica *Mixture-of-Experts* (MoE), um método que divide o processamento entre várias sub-redes, ativando apenas as mais relevantes para cada tarefa. Através destas estratégias o DeepSeek apresenta custos associados ao treino muito inferiores aos de outros LLMs, assim como uma redução significativa de recursos computacionais necessários, que se traduz na redução de emissões de carbono. A empresa divulgou que o DeepSeek-V3 (uma versão anterior do R1) foi treinado por \$6M, comparado com os \$100M para o GPT-4, utilizando apenas 2000 unidades de processamento gráfico (GPUs), enquanto que outros LLMs das maiores empresas de AI necessitam de até 16000 GPUs [77], [76], [78].

Apesar do reduzido estudo literário sobre o DeepSeek, devido ao caráter recente do modelo, investigadores acreditam que pode atingir resultados semelhantes ou ainda superiores ao do ChatGPT. Aliado à disponibilidade aos utilizadores e ao reduzido consumo de energia, tópico de debate no que toca aos outros modelos, o número de utilizadores do DeepSeek já atinge valores recorde, com potencial para se tornar um pilar na área de IA.

2.10. Prompting

Os LLMs dependem significativamente da qualidade e estrutura dos *prompts* para produzir respostas eficazes e coerentes. O *prompting* refere-se ao processo de fornecer instruções textuais ao modelo para orientar a geração de respostas. No contexto da Inteligência Artificial generativa, um *prompt* não é apenas um simples comando, mas sim um meio de comunicação que influencia diretamente a precisão e relevância das respostas geradas.

Para maximizar a eficácia do *prompting*, é essencial considerar fatores como a sintaxe, o conteúdo e a capacidade do LLM. Pequenas variações no texto de um *prompt* podem afetar drasticamente os resultados, um fenómeno conhecido como *prompt brittleness*. Esta

sensibilidade do *prompt* pode ser vista, em especial, nos verbos utilizados no pedido, por exemplo “determina”, “calcula” e “resolve” podem conduzir a respostas diferentes em certas situações [79]. Acredita-se que a criação de um padrão de escrita por parte do utilizador pode ajudar a combater esta sensibilidade, ao treinar o LLM a um determinado tipo de escrita. Outro aspeto crucial do *prompting* é a adaptação do modelo ao domínio específico da tarefa. Se um LLM não possuir conhecimento suficiente sobre um tema nos seus parâmetros de treino, um *prompt* detalhado e bem estruturado pode compensar essa falta, fornecendo informações relevantes antes de solicitar uma resposta. O uso de listas numeradas, perguntas estruturadas e exemplos específicos num *prompt*, demonstra ser eficaz para reduzir ambiguidades e melhorar a consistência das respostas. O efeito destes exemplos é observado em detalhe nas diferenças entre os *prompts* dados ao ChatGPT na Figura 11.

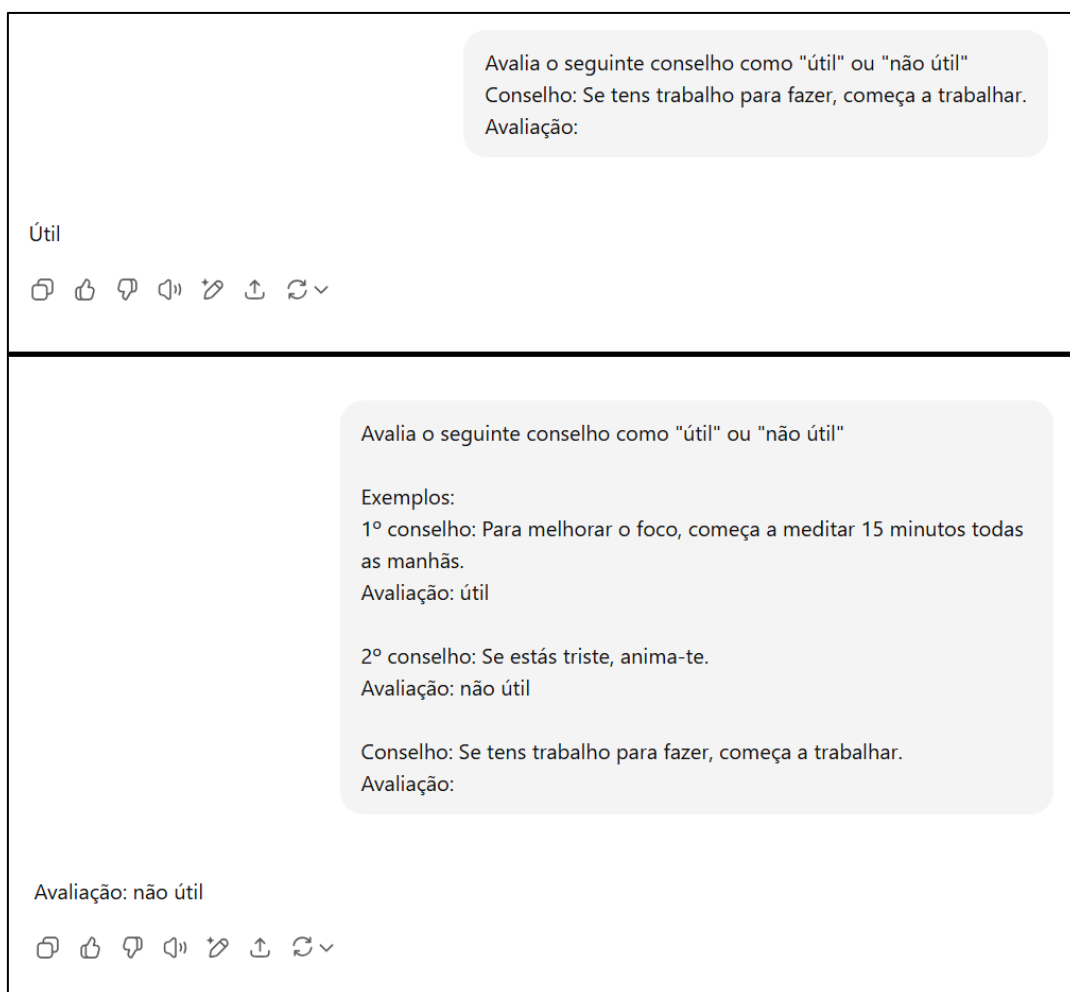


Figura 11 - Diferença entre *prompts* simples e complexos no ChatGPT

Para lidar com tarefas mais complexas são necessários métodos avançados de engenharia de *prompts*, isto é, o processo de adaptar as instruções dadas ao LLM de forma a encaminhar o seu comportamento para um objetivo específico, providenciando respostas mais coerentes e precisas [80]. A engenharia de *prompts* pode ser entendida não só como um processo técnico, mas também como uma nova forma de interação entre humano e IA, que combina conhecimento sobre o funcionamento interno dos modelos e competências de comunicação.

Estes são alguns dos métodos de engenharia de *prompts* mais eficazes, aplicados por utilizadores mais experientes, mas que podem até ser aplicados inconscientemente por qualquer pessoa:

- **Prompt chaining:** envolve a conexão de múltiplos *prompts* seguidos, cada um a utilizar a informação da resposta anterior como base (ex.: primeiro pedir para gerar um resumo de um artigo e depois pedir para reformular esse resumo numa linguagem mais simples) [79];
- **Multi-stage prompting:** semelhante ao método anterior, mas usado para dividir uma tarefa complexa em várias fases (ex.: primeiro organizar as ideias para escrever um email formal, depois gerar um rascunho de email e por último verificar a ortografia e gramática) [81];
- **Chain-of-Thought prompting:** foca-se em gerar um raciocínio passo a passo dentro de um único *prompt*, incentivando o LLM a explicar o seu raciocínio e apresentando respostas mais corretas (ex.: resolver um problema de matemática mostrando cada etapa do cálculo) [82];
- **Few-shot prompting:** consiste em adicionar alguns exemplos de tarefas semelhantes, criando um modelo, para guiar o comportamento do modelo e atingir melhores resultados (ex.: antes de pedir a tradução de uma frase para outra linguagem, apresentar outras frases com as respetivas traduções) [83];
- **Retrieval-Augmented Generation:** primeiro são recolhidas fontes de informação confiáveis, pelo utilizador ou pelo LLM, que serão utilizadas como contexto para gerar respostas mais corretas (ex.: pedir para resumir artigos académicos recentes sobre um tema – recolha automática por parte do LLM, ou apresentar um excerto de um artigo académico para resumir – recolha por parte do utilizador) [84];

Vogelsang [85] apresenta um modelo simples que pode ser utilizado de forma genérica para melhorar os *prompts* dos utilizadores em apenas 4 passos:

1. **Começar de forma simples**, com um *prompt* básico, que será desenvolvido;
2. **Apelo à ação**, começando o *prompt* com um verbo como “escreve”, “resume” ou “explica”, em vez de perguntar “podes...?” ou “consegues...?”;
3. **Adicionar contexto** específico e relevante à tarefa;
4. **Adicionar expectativas** claras do conteúdo da resposta.

Por último é importante referir que a escolha dos melhores *prompts* a usar pode mudar drasticamente com as alterações e atualizações dos LLMs, sendo necessário realizar regularmente avaliações da estrutura dos *prompts*. Através de processos demorados de tentativa e erro, especialistas de engenharia de *prompts* conseguem criar listas de instruções a seguir para realizar certas tarefas de forma otimizada em LLMs [79], [86].

2.11. Análise Bibliométrica

Para compreender a interseção entre o Escalonamento das Operações e os Large Language Models (LLMs), é essencial analisar as contribuições científicas relacionadas a esses tópicos. Este processo permite não só identificar padrões e lacunas na literatura existente, mas também compreender tendências emergentes que podem influenciar o desenvolvimento deste trabalho. Desta forma, é possível posicionar a investigação no contexto atual, construindo uma base sólida para explorar aspetos ainda pouco desenvolvidos ou desconhecidos.

A Análise Bibliométrica constitui uma abordagem sistemática e rigorosa para avaliar o estado da arte de um tema específico. Foram recolhidos dados de publicações que exploram tanto o Escalonamento das Operações como a aplicação de LLMs, a partir da base de dados *Web of Science* [87]- uma das maiores plataformas de artigos académicos, garantindo a relevância dos dados analisados.

Para tal, foi necessário primeiro definir a *query* utilizada para a recolha dos dados. Esta pesquisa focou-se principalmente em encontrar publicações que abordam especificamente os Large Language Models, logo, foram introduzidos termos como “LLMs” e “Natural Language Processing”, e foram também introduzidos alguns exemplos de modelos disponíveis atualmente, como “ChatGPT”, “Gemini”, “Copilot” e “Bard”. Depois adicionou-se o termo “*Scheduling*” para estreitar a pesquisa aos problemas de Escalonamento. Na Tabela 3 está representada a *query* completa, com o número de publicações correspondentes, no final de dezembro de 2024.

Tabela 3 - Pesquisa de palavras-chave no Web of Science

Base de Dados	Query	Resultados
Web of Science	TOPIC ("Large Language Models" OR "Large Language Model" OR "ChatGPT" OR "Gemini" OR "Copilot" OR "Bard" OR "LLM" OR "LLMs" OR "Natural Language Processing") AND "Scheduling"	250

Através da informação obtida no *Web of Science* sobre estas publicações, foram feitas várias análises quantitativas, nomeadamente à frequência de publicações por ano, às áreas de investigação mais comuns e às palavras-chave mais utilizadas.

A primeira análise apresentada, na Figura 12, mostra a evolução temporal das publicações que abordam este tema, entre 2012 e 2024. É evidente que, devido ao carácter recente dos tópicos sobre LLMs, mais de 50% das publicações foram feitas nos últimos três anos, especialmente em 2024, que viu um enorme crescimento nesta área, com 75 publicações. De facto, antes de 2015 este tema quase nunca era abordado, o que vai de acordo com o estudo feito na Revisão Bibliográfica deste trabalho, relativamente aos grandes desenvolvimentos de Deep Learning da última década e à criação do modelo *Transformer* em 2017.

Revisão Bibliográfica

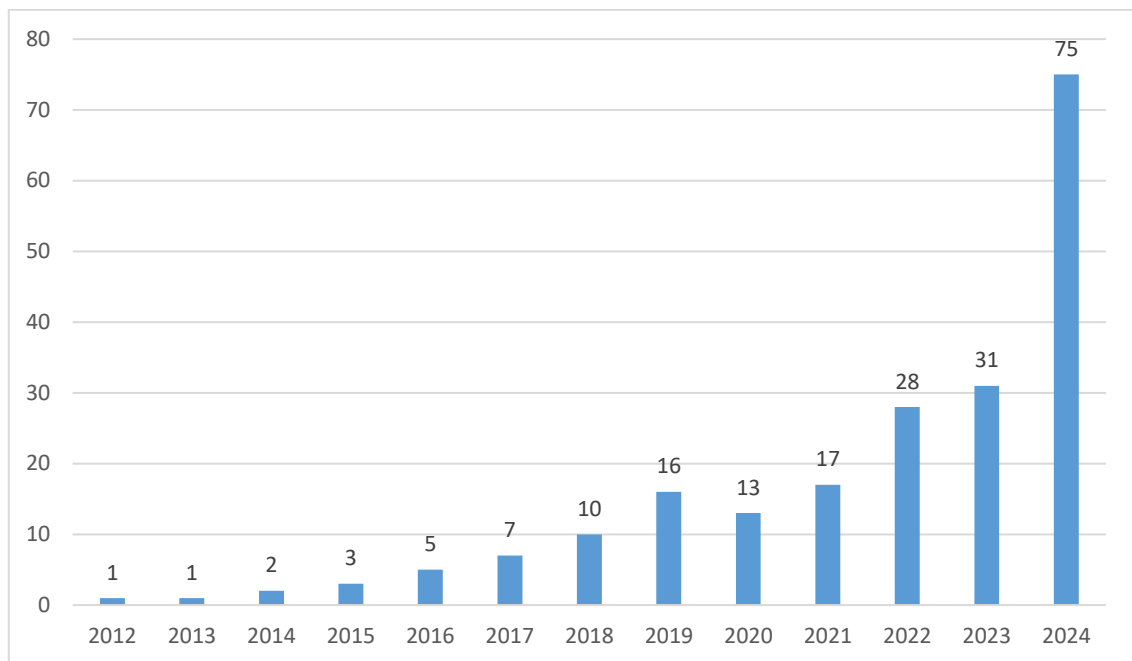


Figura 12 - Distribuição temporal de publicações entre 2012 e 2024

A análise das áreas de investigação em que as publicações se inserem também apresenta informação útil para contextualizar este trabalho. Pela Figura 13 é possível identificar a área mais comum como sendo Ciência da Computação – Métodos Teóricos, o que pode indicar que este tema apresenta uma vertente teórica substancial, para investigação e desenvolvimento. Outros campos frequentes que se pode identificar são a Engenharia Elétrica e Eletrónica, e Ciência da Computação – Inteligência Artificial, que representam um lado mais prático dos problemas.

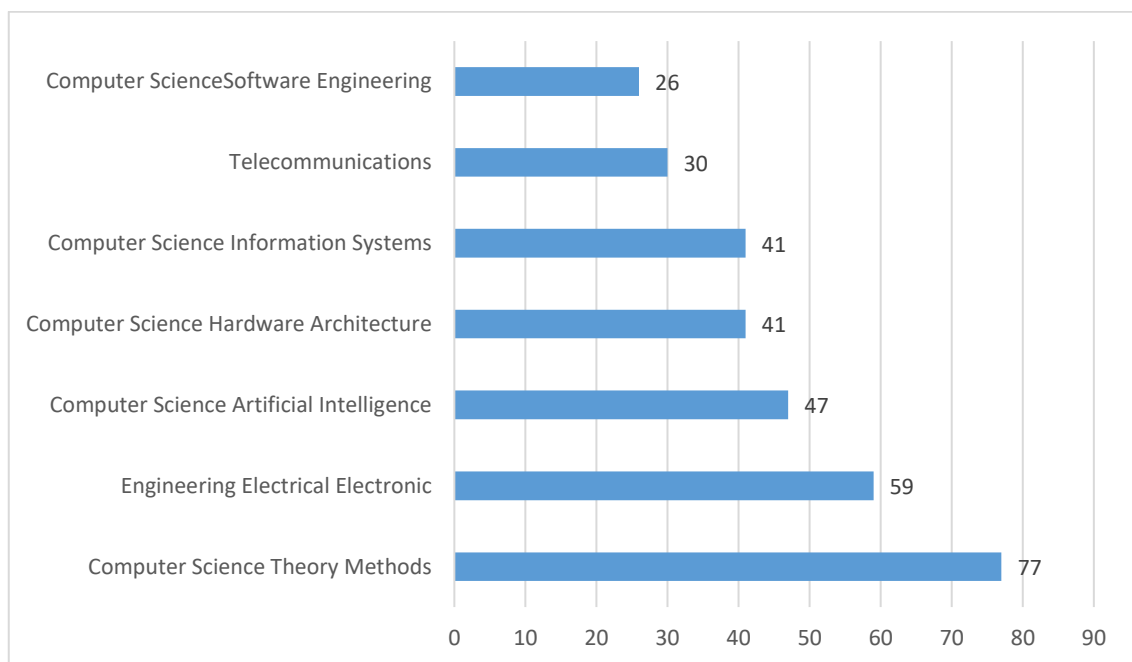


Figura 13 - Distribuição de publicações por áreas de investigação

2.11.1. Revisão do estado da arte

Caso de estudo 1 – “Investigating the Potential of Using Large Language Models for Scheduling” [88]

O caso de estudo analisa a aplicação de Large Language Models (LLMs) na resolução de problemas de escalonamento, especificamente no contexto de alocação de apresentações em conferências. Este problema é caracterizado pela necessidade de atribuir as apresentações a sessões pré-determinadas, respeitando restrições como o tema da sessão.

Uma das estratégias aplicadas foi a utilização de aprendizagem zero-shot, em que os LLMs, sem necessidade de treino adicional, avaliam a correlação entre apresentações e sessões com base na sua descrição. Isto é possível devido ao pré-treino extensivo dos LLMs em grandes volumes de dados, permitindo-lhes identificar mais corretamente semelhanças e padrões. Além disso, o estudo integrou técnicas de programação inteira para garantir que as restrições específicas do problema, como a distribuição equilibrada de trabalhos por sessão, fossem respeitadas. Esta combinação de métodos aproveita a eficiência dos LLMs na compreensão de textos e a capacidade dos modelos matemáticos na garantia de soluções viáveis.

Os resultados deste estudo mostram que a utilização de GPT-4 apresenta um desempenho superior ao método tradicional –*term frequency–inverse document frequency* (TFIDF). No entanto, foram identificadas algumas dificuldades do GPT-4 em lidar com restrições complexas e com escalabilidade para grandes volumes de dados, levando à redução da homogeneidade nestes casos. A integração de programação inteira veio trazer uma solução de otimização tanto para a aplicação de TFIDF como para o LLM utilizado. Os valores de completude e de homogeneidade da aplicação de TFIDF e da aplicação de GPT-4, com e sem programação inteira, estão apresentados na Tabela 5 e Tabela 4, respetivamente.

Tabela 4 - Resultados médios para TFIDF e LLM

Métodos	Pontuação de Completude	Pontuação de Homogeneidade
TFIDF (título e resumo)	0,37	0,20
LLM (título)	0,41	0,22

Tabela 5 - Resultados médios para programação inteira com TFIDF e LLM

Métodos	Pontuação de Completude	Pontuação de Homogeneidade
TFIDF (título e resumo)	0,42	0,42
LLM (título)	0,41	0,41

Concluindo este estudo, pode ser inferido que LLMs têm um desempenho promissor na criação de rascunhos de escalonamento, gerando soluções iniciais que podem ser refinadas, sendo interessante aplicar uma colaboração entre LLM e humano como uma primeira fase, antes da aplicação de outra técnica mais tradicional. O estudo destaca o potencial dos LLMs para acelerar o processo de escalonamento e reduzir o esforço, especialmente em cenários onde soluções aproximadas são aceitáveis. Para situações que requerem maior esforço por parte do modelo de linguagem, uma possível solução é a combinação com solucionadores numéricos.

Caso de estudo 2 – “Research on Multi-objective Optimal Scheduling Model of Industrial Text Information Based on Machine Learning” [89]

O caso de estudo analisa a aplicação de técnicas de LLMs e Machine Learning para resolver problemas de Escalonamento em ambientes de produção Flow Shop flexíveis. O objetivo principal é a otimização de múltiplos critérios, como reduzir o Tempo Máximo de Conclusão e o Consumo Energético, utilizando o algoritmo NSGA-II em conjunto com uma arquitetura de Sistemas Ciberfísicos (CPS).

Neste caso de estudo o processo apresenta 20 tarefas, cada uma com 3 operações, a serem processadas em 11 máquinas. Inicialmente este ambiente de produção foi modelado de forma a obter os tempos de processamento de cada operação.

A abordagem proposta utiliza técnicas de PLN para extrair informações essenciais, como tempos de processamento e estados das máquinas, que foram usados como entrada para o modelo de Escalonamento. Verificou-se que ao minimizar o Tempo Máximo de Conclusão das Operações, o Consumo Energético aumentava consideravelmente, o que está representado na Figura 15.

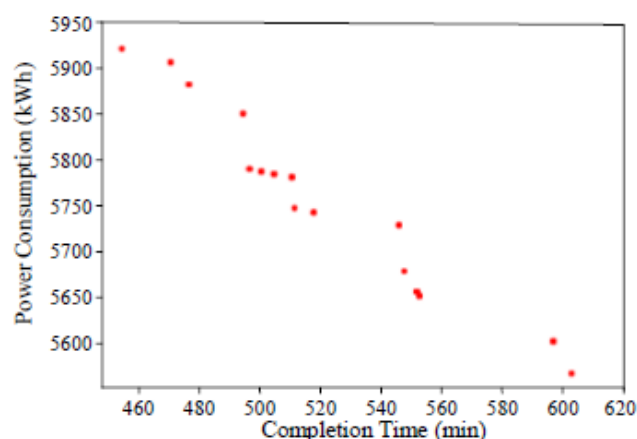


Figura 15 - Relação Tempo Máximo de Conclusão / Consumo Energético

De forma a encontrar soluções que equilibram estes dois parâmetros, foi usado o algoritmo NSGA-II para uma população de 100 (100 possíveis combinações de valores para Tempo Máximo de Conclusão e Consumo Energético), repetido 100 vezes. Como é impossível obter o mínimo valor de ambos os parâmetros ao mesmo tempo, é apresentado, na Figura 16, dois gráficos de convergência para ambos os parâmetros.

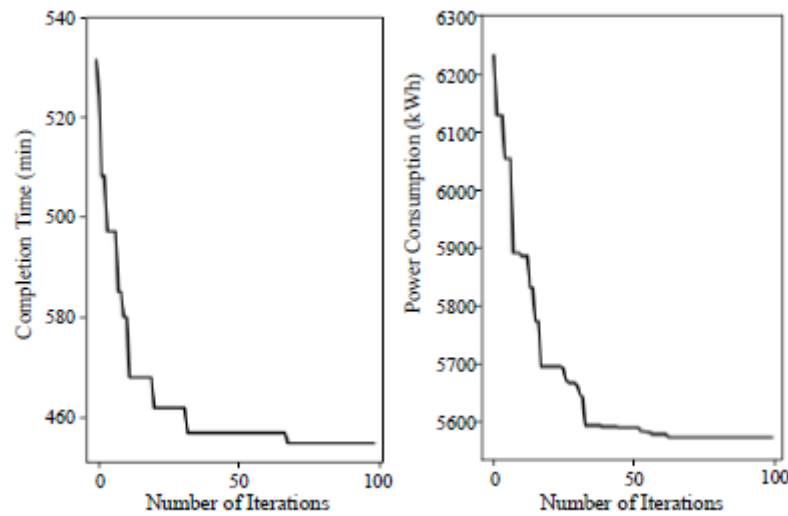


Figura 16 - Curvas de convergência do algoritmo

No final, os resultados obtidos mostram que o modelo proposto conseguiu diminuir o Tempo Máximo de Conclusão das Operações para 454 minutos e o Consumo Energético total para 5934,7 kWh, representando ganhos significativos na eficiência do processo.

Apesar do sucesso alcançado, o estudo identificou desafios na escalabilidade da solução quando aplicada a sistemas mais complexos e na necessidade de infraestrutura computacional robusta para lidar com grandes volumes de dados. Contudo, os resultados destacam o potencial da integração de técnicas de PLN e algoritmos de otimização para melhorar os problemas de Escalonamento das Operações.

Caso de estudo 3 – “CAPP-GPT: A Computer-Aided Process Planning-Generative Pretrained Transformer Framework for Smart Manufacturing” [90]

Este caso de estudo explora a integração de LLMs em sistemas de manufatura inteligente, com o objetivo de otimizar o planejamento de processos e o Escalonamento de Operações. Este trabalho destaca-se por abordar a crescente complexidade de sistemas industriais, utilizando tecnologia de ponta para superar desafios como a adaptação a interrupções não planejadas e a necessidade de maior flexibilidade no Escalonamento.

A abordagem proposta combina técnicas de Processamento de Linguagem Natural e modelos de Deep Learning baseados em *Transformers* para interpretar dados técnicos e transformá-los em planos de processos detalhados. Este processo é suportado por *Digital Twins* - representações digitais de sistemas ou processos físicos que monitorizam operações em tempo real e ajustam parâmetros conforme necessário. O uso de algoritmos de otimização, como o NSGA-II, é também destacado, permitindo lidar com múltiplos objetivos em simultâneo, como minimizar custos e maximizar a eficiência.

Durante um caso de teste, foram definidas 21 operações possíveis de serem feitas a uma peça simples. Foram definidas, depois, 8 variantes que possuem um determinado número destas operações. O objetivo é encontrar uma solução para a ordem de conclusão das operações que

minimize os custos de produção. Os resultados apresentados foram promissores, com melhorias significativas na eficiência e na capacidade de adaptação em ambientes dinâmicos. Neste cenário de teste com 33 soluções diferentes, o custo mínimo de produção alcançado foi de \$41697,8, enquanto as adaptações em tempo real reduziram falhas e otimizaram a alocação de recursos. No entanto, o estudo identifica desafios, comuns a outros casos de estudo relevantes ao tópico, como a escalabilidade para cenários mais complexos e a necessidade de infraestrutura computacional robusta.

Em conclusão, este caso de estudo ilustra o potencial de LLMs e técnicas de PLN na melhoria do escalonamento de operações industriais. A integração dessas ferramentas permite maior precisão no planejamento, maior flexibilidade operacional e contribui para o avanço das práticas na manufatura inteligente.

Outros casos de estudo relevantes

Tabela 6 - Outros Casos de Estudo Relevantes

Autor	Caso de estudo	Conclusões
Zheng et al. [91]	O estudo propõe um canal de inferência para LLMs baseada no escalonamento de sequências, otimizando recursos computacionais com técnicas como agrupamento por comprimentos de respostas	Aumento de 86% na eficiência sem comprometer a qualidade
Pietro et al. [92]	O estudo avalia o uso do ChatGPT para gerar cronogramas de construção para um projeto simples	Potencial para acelerar processos preliminares, embora com limitações em precisão e detalhe
Kwan, Ho Y. [93]	O estudo desenvolve um sistema de telemedicina baseado em LLMs, para melhorar o agendamento, teleconsultas e suporte clínico	Maior taxa de comparecimento (15,38%) e alta satisfação dos utilizadores
Wu et al. [94]	O estudo propõe uma estrutura para adaptar LLMs a 3 situações de <i>networking</i> – previsão de visualização num vídeo 360º, qualidade de <i>streaming</i> adaptativa, e escalonamento de tarefas em <i>clusters</i> de computadores	Redução de erro médio. Melhoria de qualidade da experiência. Redução do tempo médio de conclusão de tarefas
Papaioannou & Doudali [95]	O estudo avalia o impacto de diferentes cargas de trabalho e memória no desempenho de sistemas de inferência com LLMs, em quatro cenários: geração de texto, sumarização, questionamento-resposta e conversação, para dois tipos de sistema: foco em baixa latência e foco em alta capacidade	Geração de texto teve desempenho superior. Otimizações de memória trazem benefícios significativos em sistemas de alta capacidade
Zhou et al. [96]	O estudo utiliza LLMs para escalonamento de chegadas de voos, aplicando modelagem de linguagem e fine-tuning supervisionado para reduzir conflitos e atrasos	Resultados promissores na redução de atrasos, ainda que próximos de métodos como FIFO
Jiang et al. [97]	O estudo propõe o SLoB, um sistema de escalonamento para empresas que usam <i>clusters</i> locais de GPU em inferência de LLMs	Melhoria da métrica de satisfação em até 29,4x, e da eficiência em 3x. Equilíbrio entre uso de memória e velocidade de processamento

2.12. Análise crítica da revisão bibliográfica

Os Large Language Models demonstraram ser ferramentas versáteis para resolver problemas de escalonamento, organização e planeamento. A capacidade destes modelos de compreender linguagem natural e gerar soluções iniciais para problemas estruturados destaca-se como um avanço significativo. Os LLMs são úteis para criar soluções rápidas e eficientes, reduzindo o esforço manual e acelerando os processos preliminares.

Contudo, a aplicação de LLMs em problemas de Escalonamento requer colaboração com métodos estatísticos e matemáticos, e com *feedback* humano, para garantir a otimização das soluções. Enquanto os LLMs são excelentes em analisar padrões e propor soluções iniciais, métodos como a programação inteira podem refinar os resultados, melhorando a sua precisão. Este trabalho salienta que os LLMs devem ser encarados como complementares, e não substitutos, de técnicas tradicionais em problemas mais complexos.

Adicionalmente, em problemas de maior escala, que envolvem conjuntos de dados extensos ou restrições muito específicas, os LLMs enfrentam desafios significativos. As limitações incluem dificuldades em garantir escalabilidade, adaptação a restrições complexas e necessidade de infraestrutura computacional elevada. Embora promissores, esses modelos ainda não possuem a robustez necessária para substituir completamente abordagens otimizadas em grandes cenários industriais.

Uma limitação encontrada na realização deste trabalho foi a falta de publicações sobre o uso de LLMs para problemas de Escalonamento das Operações em ambientes industriais. Esta escassez deve-se, em parte, ao caráter recente dos LLMs, cujo desenvolvimento e aplicação prática só começaram a ganhar destaque nos últimos anos. Ainda assim, o potencial destes modelos é mais evidente em estudos em áreas como a organização de tarefas na área medicinal, otimizando fluxos de trabalho em telemedicina, e na alocação de recursos computacionais, como em redes de computadores e agendamento de cluster. Estes estudos fornecem informações valiosas que podem ser adaptadas e aplicadas ao Escalonamento das Operações em contextos industriais.

3. Métodos e Aplicações

A otimização do Escalonamento das Operações é um procedimento dinâmico que requer muita atenção, devido ao elevado número de variáveis inerentes dos problemas. Existem vários métodos de otimização que podem ser utilizados nos problemas de Escalonamento das Operações, cujos contextos de aplicação variam de acordo com os requisitos e as restrições de cada situação, sendo a escolha destes um processo fundamental na melhoria dos sistemas de produção.

Uma alternativa que tem surgido na otimização do Escalonamento das Operações é o uso de *Large Language Models* (LLMs), que apresentam uma capacidade de interpretação e resposta incrivelmente rápida. É importante, no entanto, compreender quais os métodos de otimização a serem utilizados como referência para o raciocínio destes modelos, e como os direcionar à utilização de certas abordagens que poderão resultar em melhores soluções.

Assim, neste capítulo são apresentados inicialmente alguns problemas de Escalonamento das Operações relevantes ao estudo, discutindo diferentes métodos de otimização que podem ser usados pelos LLMs, e a estratégia de *prompting* utilizada neste trabalho. O objetivo é avaliar o desempenho dos LLMs na resolução de problemas de Escalonamento das Operações.

3.1. Problemas de Escalonamento das Operações

Com o objetivo de tentar replicar ambientes industriais complexos, este trabalho vai incidir sobre os problemas de Escalonamento das Operações em *Job Shop*.

Neste tipo de problemas existem m máquinas que podem processar qualquer tarefa, sendo estas compostas por várias operações e tendo, cada uma, um percurso pré-determinado através das máquinas. Habitualmente, numa *Job Shop*, as tarefas possuem $t = m$ operações, o que implica que cada operação é atribuída a uma máquina diferente, levando a que a tarefa percorra todas as máquinas ao longo do seu processamento. É possível utilizar a mesma formulação para situações em que uma tarefa volta a visitar uma máquina para processamento de uma operação distinta ou quando não passa por algumas máquinas [9].

Os problemas em *Job Shop* são então considerados *NP-hard*, isto é, são problemas que com o aumento do tamanho, mais tarefas para processar em mais máquinas, torna-se exponencialmente mais complicado aplicar um algoritmo eficiente que os consiga resolver num período de tempo razoável. Isto é, quanto maior o problema fica, mais difícil torna-se até verificar se uma determinada solução está correta.

De seguida é apresentada uma instância de Escalonamento das Operações em *Job Shop*, descrita por [9] e com a formulação baseada nos trabalhos de Pinedo [13] e de Fernandes & Lourenço [98], com o objetivo de minimizar o *makespan* ($J_m | C_{max}$).

Em problemas desta natureza é comum representar a informação conhecida, que não se altera durante a resolução do problema, como os seguintes parâmetros principais:

- m máquinas pertencentes a $P = \{P_1, P_2, \dots, P_m\}$;
- n tarefas pertencentes a $J = \{J_1, J_2, \dots, J_n\}$;
- Cada tarefa possui uma sequência específica de processamento de h operações pertencentes a $O = \{O_{1,j}, O_{2,j}, \dots, O_{h,j}\}$;
- $O_{i,j}$ representa a operação i da tarefa j , processada numa máquina m , com um tempo de processamento $p_{i,j}$.

Quanto às variáveis de decisão – dados que se pretende determinar, consideram-se:

- O tempo de início da operação i da tarefa j , ($t_{i,j}$).
- O *makespan* ($C_{máx}$), objetivo a determinar.

A formulação seguinte descreve a função objetivo e as restrições associadas a este problema:

F.O.

$$\min C_{máx} = \min[\text{máx}(t_{h,n} + p_{h,n})] \quad (4)$$

S.A.

$$t_{i,j} + p_{i,j} \leq t_{(i+1),j} \quad , \forall j \in J, \forall i \quad (5)$$

$$t_{i,j} + p_{i,j} \leq t_{k,l} \vee t_{k,l} + t_{k,l} \leq t_{i,j} \quad , \forall O_{i,j}, O_{k,l} \text{ em } m \quad (6)$$

$$t_{i,j} \geq 0 \quad , \forall O_{i,j} \quad (7)$$

A equação (4) apresenta o objetivo deste problema: minimizar o *makespan*, o que pode ser representado pela minimização do tempo de conclusão da última operação (h) da última tarefa (n) (tempo de início da operação + tempo de processamento da operação).

De seguida é necessário adicionar restrições. Para assegurar que a sequência de processamento das operações das tarefas é cumprida, é introduzida a equação (5). Esta restrição vai obrigar a operação $O_{(i+1),j}$ a apenas iniciar o processamento quando a operação anterior $O_{i,j}$ terminar.

Já a equação (6) é usada para evitar a sobreposição de operações na mesma máquina, isto é, duas operações a serem processadas ao mesmo tempo na máquina m . Para tal, deve-se assegurar que $O_{k,l}$ só inicia o seu processamento quando $O_{i,j}$ estiver completa, ou vice-versa, assim permitindo a decisão de qualquer ordem entre operações de tarefas diferentes.

Por último a equação (7) limita os tempos de início de processamento das operações a valores maiores ou iguais a zero – restrição de não negatividade.

Com a formulação definida, procede-se para os dados específicos de uma instância. Nesta instância estão disponíveis 3 máquinas ($m = 3$) para processar 4 tarefas ($n = 4$), cada uma com 3 operações, estando pré-definida a sequência de processamento destas operações. Os valores indicados na Tabela 7 referem-se aos tempos de processamento de todas as operações.

Tabela 7 - Tempos de processamento de um problema exemplo em Job Shop com 4 tarefas e 3 máquinas

	Tarefa 1		Tarefa 2		Tarefa 3		Tarefa 4	
	$O_{i,j}$	$p_{i,j}$	$O_{i,j}$	$p_{i,j}$	$O_{i,j}$	$p_{i,j}$	$O_{i,j}$	$p_{i,j}$
Máquina 1	1	4	2	4	3	3	3	1
Máquina 2	2	3	1	1	2	2	1	3
Máquina 3	3	2	3	4	1	3	2	3

Como discutido anteriormente, pretende-se definir a ordem das operações em cada máquina, de forma a reduzir o tempo total de conclusão das operações. Após serem aplicados os métodos de otimização necessários, foi encontrada a solução mais apropriada para o caso, apresentada na Figura 17 através de um gráfico de Gantt. Um gráfico de Gantt utiliza uma escala temporal para representar as operações, através de barras com indicação do tempo de processamento, tendo em conta a disponibilidade de cada máquina e as restrições de precedência. Este tipo de representação é muito utilizado em equipas de gestão de projetos para se concentrarem no que precisa de ser feito, e quando precisa de ser feito [99].

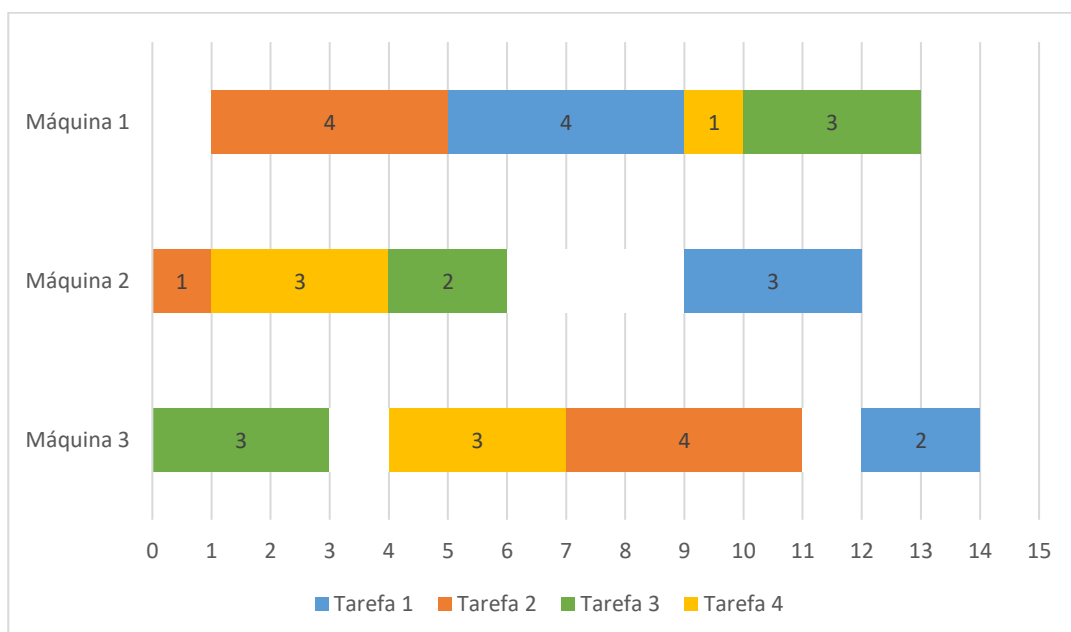


Figura 17 - Gráfico de Gantt do problema exemplo em Job Shop

Outra forma de representação dos problemas em *Job Shop* é através de um grafo disjuntivo. Estes modelos apresentam vários nós simbolizando as operações do problema, interligadas por arestas, e compreendidas entre dois nós (de início e de fim do processo). Cada aresta está acompanhada do tempo de processamento da operação do nó inicial, e, seguindo até ao nó final, representa a sequência das operações pertencentes à mesma tarefa. Podem ainda ser representadas as arestas que ligam as operações que necessitam de ser processadas da mesma máquina. Na Figura 18 observa-se o grafo disjuntivo para a mesma solução desta instância.

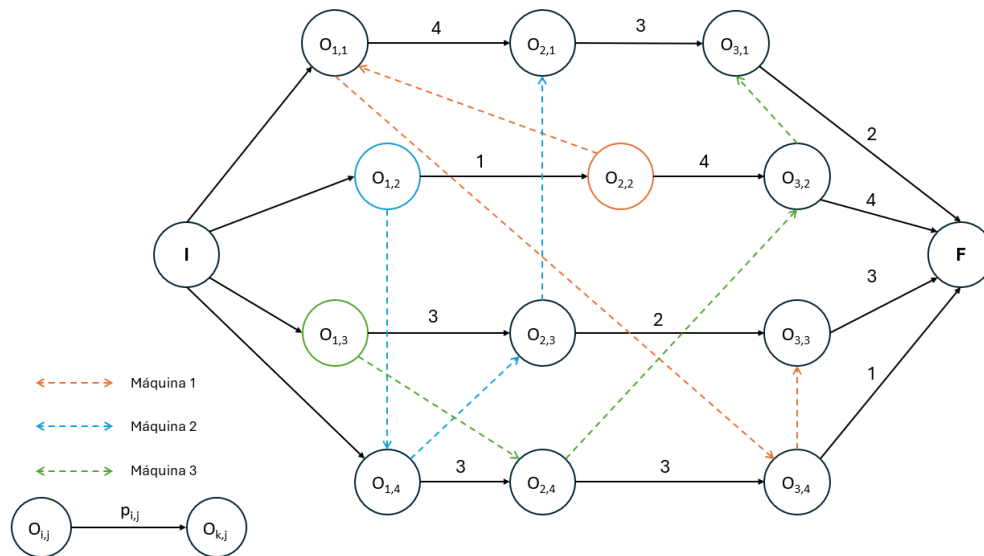


Figura 18 - Grafo Disjuntivo do problema exemplo em *Job Shop*

Através da solução representada nestes dois modelos, é possível concluir todas as operações em 14 u.t. (unidades de tempo). Foram verificadas as restrições de precedência e de sobreposição, comprovando-se que todas as tarefas seguem a sequência pré-determinada, e que nenhuma máquina processa duas operações ao mesmo tempo em nenhum momento.

Conclui-se que os problemas em *Job Shop* representam desafios complexos de escalonamento das Operações, onde, apesar de existirem múltiplas soluções possíveis, o objetivo é encontrar a que otimiza determinado(s) critério(s), como o *makespan*. No entanto, à medida que a dimensão e a complexidade do problema aumentam, torna-se significativamente mais difícil encontrar a solução ótima com métodos tradicionais de otimização, tornando-se necessário aplicar métodos mais avançados de otimização, abordados no subcapítulo seguinte.

3.2. Métodos de otimização

A resolução dos problemas de Escalonamento envolve métodos de otimização que podem ser classificados entre os métodos exatos e os métodos aproximados. Os primeiros têm a capacidade de explorar por completo o espaço de soluções, garantindo um Escalonamento ótimo, sendo ideais para situações em que a precisão é crítica e o problema é relativamente pequeno ou bem estruturado, enquanto os aproximados focam-se em encontrar uma solução “boa” num espaço de tempo mais curto, tornando-os adequados em contextos onde a rapidez é essencial.

Nesta secção são apresentados os métodos mais comuns para problemas de Escalonamento, destacando aqueles que os LLMs tendem a utilizar mais frequentemente.

Métodos Exatos

A resolução exata envolve duas componentes essenciais: a formulação matemática do problema através de métodos específicos e a aplicação de técnicas de resolução para estes modelos.

Modelos Matemáticos de Formulação

Mixed Integer Linear Programming (MILP)

Através de modelos matemáticos compostos por variáveis de decisão (contínuas e inteiras), uma ou mais funções objetivo e restrições representadas por equações ou inequações lineares, é possível formular um problema de Escalonamento. Os modelos MILP são frequentemente resolvidos através de técnicas avançadas com o algoritmo *Simplex* (para a parte linear do problema) e o *Branch and Bound* (para as variáveis inteiras), e aplicados em *solvers* como o CPLEX ou o Gurobi para obter soluções exatas em problemas de pequena-média complexidade. Para instâncias maiores, típicas de contextos industriais, os modelos MILP tornam-se frequentemente impraticáveis devido ao elevado número de variáveis e restrições, que exigem tempos de resposta longos e recursos computacionais significativos [100], [101].

Em relação à aplicação integrada de MILP com LLMs, foi perguntado ao ChatGPT se é possível utilizá-lo para encontrar soluções otimizadas em problemas de Escalonamento em Job Shop, obtendo a resposta apresentada na Figura 19. Compreende-se que o ChatGPT, tal como a maioria dos modelos de linguagem, não tem a capacidade de executar diretamente o código necessário para resolver o modelo de MILP, mas consegue gerá-lo e apresentar ao utilizador.

✅ O que o ChatGPT Consegue Fazer:

1. Modelar o teu problema como MILP:
 - Definir variáveis binárias para o sequenciamento das operações.
 - Criar variáveis para os tempos de início, restrições de precedência e de conflito de máquina.
 - Formular a função objetivo para minimizar o makespan (tempo total).
2. Escrever o código MILP:
 - Em CPLEX OPL, Python com PuLP, Gurobi Python API, ou Pyomo, por exemplo.
 - Gerar um modelo MILP completo e correto a partir dos dados dos teus jobs e máquinas.
3. Ajudar a interpretar soluções:
 - Analisar os resultados e transformá-los em gráficos de Gantt, tabelas de operações ou resumos.
 - Explicar quais restrições estão ativas, se a solução é ótima, etc.

❌ O que o ChatGPT Não Consegue Fazer Sozinho:

1. Executar o solver MILP internamente:
 - O ChatGPT não tem acesso direto a solvers como CPLEX, Gurobi ou CBC durante a execução.
 - Não consegue "resolver" numericamente o modelo MILP dentro desta interface.
2. Executar código externo arbitrário:
 - Terás de copiar o modelo gerado e executá-lo no teu computador com o solver apropriado.

Figura 19 - Resposta de ChatGPT sobre aplicação de MILP

Constraint Programming (CP)

Constraint Programming, ou Programação por Restrições, é utilizada de uma forma abrangente devido à facilidade em incorporar restrições diretamente na formulação do problema. Este método envolve a definição explícita das variáveis, os seus domínios, e das restrições que regem as interações entre elas, de forma a estruturar devidamente o problema, para começar a exploração sistemática do campo de soluções. CP utiliza uma técnica eficiente para alcançar uma solução viável que satisfaça todas as restrições: a propagação de restrições, na qual são analisados todos os valores possíveis para as variáveis, e eliminados aqueles que não estão dentro dos requisitos.

Em comparação com os modelos MILP, *Constraint Programming* oferece maior flexibilidade para aspetos mais complexos em ambiente industrial, como os tempos de *setup* dependentes da sequência e períodos de indisponibilidade das máquinas. Esta característica possibilita, segundo alguns resultados experimentais [101], a obtenção de soluções ótimas mais rapidamente do que modelos MILP, para instâncias pequenas-médias, mas podem perder eficiência em instâncias maiores. Observa-se que, devido às técnicas de propagação de restrições usadas em CP, estes modelos têm dificuldades a resolver problemas com um número excessivo de soluções, sendo frequentemente utilizados em problemas de Escalonamento de menor dimensão [101], [102].

Modelos CP podem ser utilizados em *solvers* como CP-SAT (através de OR-Tools) e CP *Optimizer*, no entanto, tal como em modelos MILP, os LLMs comuns não têm capacidade de processar o código necessário para resolver problemas diretamente, apenas para o gerar, conforme pode ser observado na Figura 20, num excerto durante o período de “pensamento” do ChatGPT ao tentar encontrar uma solução para um problema de Escalonamento específico.

- Parece que o ambiente onde estou trabalhando não possui o OR-Tools acessível. Embora tenha tentado importar o OR-Tools no ambiente de Python visível para o usuário, ele não estava disponível. Talvez no ambiente de análise ele esteja, mas, ao testar, ficou claro que não existe. Isso significa que não podemos calcular o ótimo numérico via código. A alternativa? Considerar uma solução aproximada, deixando claro que o CP-SAT poderia ser usado para estimar o makespan, com base em fontes.

Figura 20 - Pensamento de ChatGPT sobre aplicação de CP

Apesar de não ser possível aplicar LLMs para obter soluções imediatas através de *Mixed Integer Linear Programming* ou *Constraint Programming*, durante o desenvolvimento deste trabalho foi utilizado o *software IBM ILOG CPLEX Optimization Studio* para obter as soluções ótimas das instâncias em estudo, através de CP.

Técnicas de Otimização

Branch and Bound

O método de *Branch and Bound* utiliza uma estratégia baseada numa árvore de decisões para explorar o espaço de soluções. O nó raiz representa o problema a ser analisado e os nós internos

representam subdivisões do problema. Esta pesquisa é feita por etapas, onde em cada etapa dá-se uma “ramificação” (*branching*), gerando soluções parciais – nós de folha, que são avaliadas durante a fase de “limitação” (*bounding*) para determinar se possuem potencial para conterem a solução ótima. Nesta fase de “limitação” é calculado um limite inferior ou superior de um valor de decisão para cada solução e são “podados” os ramos que não apresentam soluções dentro desses valores, reduzindo assim o espaço de pesquisa e levando a soluções cada vez mais completas [103]. Na Figura 21 observa-se um exemplo de um algoritmo *Branch and Bound*.

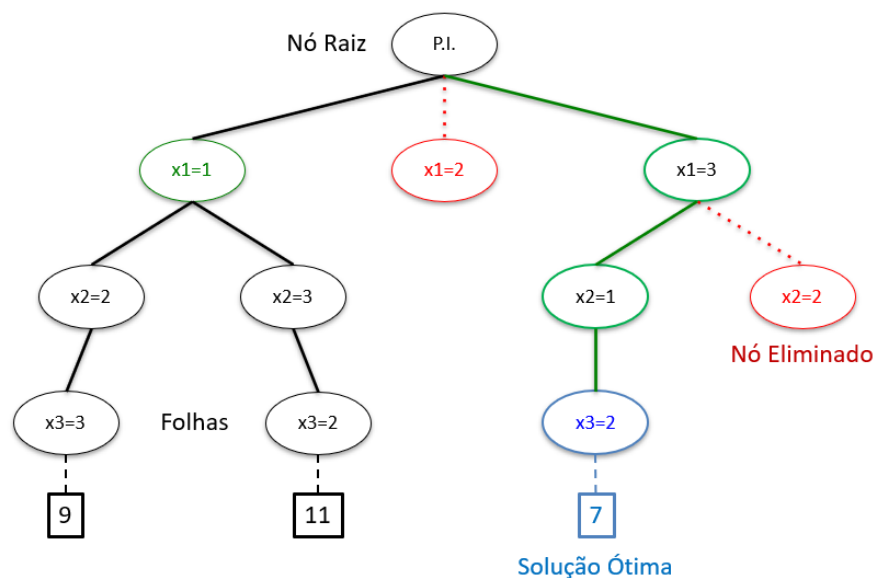


Figura 21 - Exemplo de um algoritmo Branch and Bound (adaptado de [104])

O método de *Branch and Bound* é particularmente eficaz na otimização de problemas de Escalonamento, especialmente em instâncias de pequena a média dimensão. Contudo, devido ao aumento exponencial do espaço de soluções, é menos viável aplica-lo a problemas de maior dimensão e complexidade [105].

Simplex

O Simplex é um algoritmo clássico para resolver problemas de programação linear, que constitui a base para resolver modelos MILP, especificamente a parte contínua destes problemas. O algoritmo funciona percorrendo os vértices (ou soluções extremas) da região viável, definida pelas restrições lineares do problema. A cada iteração o algoritmo seleciona um vértice adjacente ao atual que melhore o valor da função objetivo, movendo-se sistematicamente através do espaço de soluções até atingir um ponto em que nenhuma melhoria adicional seja possível, indicando a obtenção da solução ótima. A Figura 22 apresenta um exemplo de representação do algoritmo Simplex. Este processo envolve ainda a formulação do problema numa tabela, onde cada coluna representa uma variável e cada linha uma restrição, assim como a linha da função objetivo. Embora em alguns casos teóricos o Simplex apresente um tempo de resolução exponencial, em casos práticos industriais é demonstra alta eficiência e facilidade de implementação.

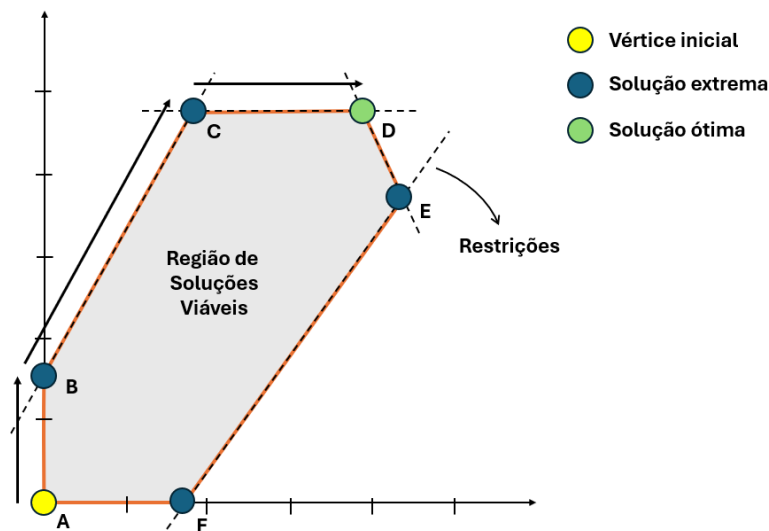


Figura 22 - Representação do algoritmo Simplex

Métodos Aproximados

Heurísticas

Em contexto industrial, especialmente em ambientes de produção complexos, pode não ser prático aplicar métodos exatos de otimização devido à elevada exigência computacional. Nestas situações os métodos aproximados, como as heurísticas, são frequentemente usados para encontrar soluções rapidamente, embora as mesmas possam não ser as soluções ótimas.

As heurísticas podem ser divididas em dois grandes grupos: as regras simples de prioridade e as heurísticas construtivas. As primeiras regem-se por sequências baseadas em critérios predefinidos e imediatos, o que conduz a uma aplicação simples e rápida, mas muito limitada. Estas heurísticas geralmente encontram soluções subótimas, devido à sua natureza rígida.

Estas são algumas das heurísticas simples mais utilizadas na otimização de problemas de Escalonamento:

- **Longest Processing Time (LPT):** Esta heurística atribui prioridade às tarefas com maior tempo de processamento, sendo especialmente eficaz em situações onde se pretende evitar atrasos significativos nas tarefas com maior duração.
- **Shortest Processing Time (SPT):** Esta heurística prioriza as tarefas com menor tempo de processamento, minimizando o tempo médio de espera e sendo muito usada em contextos onde a rapidez média de execução das tarefas é crucial.
- **Earliest Due Date (EDD):** Prioriza as tarefas com datas de entrega mais próximas, sendo particularmente relevante em ambientes onde a satisfação dos prazos de entrega é importante.
- **First In First Out (FIFO):** Esta regra muito simples atribui prioridade às tarefas conforme a ordem de chegada, sendo usada como padrão em muitos ambientes de produção que não utilizam outras regras de Escalonamento.

Estas regras são também consideradas “gulosas”, ou seja, tomam decisões passo-a-passo selecionando sempre a opção mais correta em cada um, sem considerar escolhas anteriores ou possíveis consequências futuras.

Já as heurísticas construtivas são adequadas para problemas mais complexos, e consistem em construir progressivamente calendários de execução das tarefas, inserindo gradualmente as operações com base numa característica do problema. No final pode ser feita uma pesquisa local para efetuar pequenas alterações com objetivo de melhorar a solução final [106].

- **NEH (Nawaz, Enscore e Ham):** Especialmente concebida para problemas do tipo Flow Shop, esta heurística gera soluções iniciais de boa qualidade ao inserir as operações de forma ordenada, frequentemente resultando numa boa aproximação à solução ótima.
- **Shifting Bottleneck:** Este método identifica iterativamente o recurso ou máquina crítica (gargalo) e otimiza sequencialmente a sua programação. Cada iteração simplifica o problema ao tratar cada recurso isoladamente, possibilitando soluções eficazes para problemas moderadamente complexos.
- **Giffler-Thompson Algorithm:** Este algoritmo constrói soluções passo a passo, gerindo conflitos e prioridades durante a construção sequencial das soluções. É particularmente conhecido pela eficácia na resolução de problemas de *Job Shop*, pela sua capacidade de proporcionar uma exploração estruturada e eficiente do espaço de soluções.

Apesar da simplicidade das heurísticas, continuam a ser métodos muito utilizados na resolução de problemas de Escalonamento, apresentando, em muitos casos, resultados satisfatórios. Devido à sua natureza determinística, os LLMs possuem a capacidade de implementar estas estratégias durante a procura de uma solução viável para os problemas, demonstrando ser, para algumas instâncias, o método mais eficaz para alcançar uma “boa” solução. Isto é possível ser identificado especialmente durante a fase de “pensamento” ou “análise” dos LLMs, onde regularmente passa por etapas de aplicação de simples regras de prioridade e, quando estas estratégias não levam a uma solução de qualidade, é de seguida aplicado uma heurística mais complexa para melhorar essa solução.

Meta-heurísticas

Em problemas de escalonamento complexos as heurísticas construtivas podem não ser capazes de encontrar soluções satisfatórias. É neste contexto que surgem as meta-heurísticas, métodos aproximados que exploram eficientemente o espaço das soluções, com o objetivo de encontrar a solução ótima ou próximas da solução ótima.

A aplicação das meta-heurísticas em problemas industriais tem sido especialmente destacada devido à sua robustez, permitindo uma adaptação a diferentes tipos de problemas, restrições e objetivos. Além disso, estas técnicas frequentemente são menos sensíveis a variações nos dados de entrada, tornando-as especialmente apropriadas para ambientes industriais, onde a incerteza e a variabilidade são frequentes.

As seguintes meta-heurísticas destacam-se como particularmente relevantes para problemas de escalonamento:

- **Genetic Algorithm (GA):** Inspirados no processo evolutivo da natureza, os Algoritmos Genéticos operam sobre um conjunto de soluções (população) utilizando mecanismos de seleção, cruzamento e mutação para criar novas soluções. A evolução contínua desta população tende a melhorar progressivamente as soluções, permitindo explorar de uma forma eficaz o espaço de pesquisa e adaptar-se rapidamente às características específicas dos problemas de Escalonamento.
- **Simulated Annealing (SA):** Inspirado no processo físico de arrefecimento de metais, este método permite, ocasionalmente, aceitar soluções inferiores com o objetivo de evitar a convergência em mínimos locais, especialmente útil em espaços de soluções complexos e irregulares. Ao longo do tempo, a probabilidade de aceitação de soluções inferiores diminui gradualmente, permitindo que o algoritmo explore de forma extensa o espaço inicial, focando-se progressivamente em soluções melhores.
- **Tabu Search (TS):** Este método utiliza uma estrutura de memória (lista tabu) para armazenar soluções recentemente visitadas, impedindo assim que o algoritmo volte a soluções anteriores e entre em ciclos repetitivos. A *Tabu Search* combina esta memória com mecanismos para explorar intensivamente soluções promissoras, destacando-se pela capacidade de escapar dos ótimos locais através da introdução de critérios flexíveis no processo de pesquisa.

Ao contrário das heurísticas construtivas, as meta-heurísticas apresentam maior dificuldade de aplicação aos LLMs comuns. Apesar destes não estarem projetados para executar diretamente algoritmos mais complexos, com as múltiplas iterações necessárias para alcançar uma solução, podem gerar um código que implementa meta-heurísticas ou até adotar uma estratégia híbrida, onde aplica inicialmente outro método de otimização apresentado anteriormente, como as heurísticas construtivas, seguido de uma simulação de um número reduzido de iterações de uma meta-heurística, como o *Genetic Algorithm*. Esta abordagem assegura uma pesquisa mais profunda do espaço de soluções, no entanto com maior dificuldade de implementação.

3.3. Preparação dos LLMs

Antes de se iniciar a aplicação prática dos *Large Language Models* (LLMs) na resolução de problemas de Escalonamento, foi necessária uma fase de preparação de modo a garantir a fiabilidade dos resultados e a coerência da metodologia. Esta fase consistiu, principalmente, na seleção dos modelos a utilizar, na definição dos critérios de teste e na estrutura dos prompts.

Foram considerados vários modelos populares, incluindo ChatGPT, Copilot, Gemini e DeepSeek. No entanto, observou-se que tanto o Copilot como o Gemini revelaram limitações significativas em problemas simples de Escalonamento, demonstrando constantemente a incapacidade de apresentar a solução para estes problemas, o que levou à sua exclusão das próximas fases do trabalho. Assim, o estudo foi centrado no ChatGPT e no DeepSeek, modelos que demonstraram maior estabilidade, capacidade de raciocínio e uma abordagem mais estruturada na resolução das tarefas propostas. É ainda importante mencionar que o DeepSeek foi lançado durante o desenvolvimento deste trabalho, tendo gerado desde o início um grande entusiasmo na

comunidade tecnológica pelas suas capacidades. Assim, criou-se um enorme interesse em comparar este modelo emergente com um modelo já estabelecido como o ChatGPT. Inicialmente os testes de comparação foram realizados entre os modelos GPT o4-mini e DeepSeek R1, que apresentam maior capacidade de raciocínio de entre os modelos disponíveis. No entanto, para analisar o potencial máximo dos LLMs na resolução de problemas de Escalonamento decidiu-se, posteriormente, utilizar também o modelo GPT o3, do ChatGPT Pro, um modelo recente que possui uma capacidade de raciocínio superior aos seus antecessores.

Para assegurar uma avaliação justa e imparcial dos modelos, removeu-se todos os dados e configurações previamente armazenadas na memória destes, criando novas contas com apenas o propósito de resolver os problemas de Escalonamento. O objetivo foi simular uma interação "em branco" e evitar que resultados fossem influenciados por interações passadas ou informações adicionais retidas na memória. Além disso, todas as interações com os modelos para resolução dos problemas foram conduzidas exclusivamente em inglês, de forma a assegurar maior consistência na interpretação das instruções, dado que a maioria dos modelos de linguagem atuais apresenta um desempenho otimizado nesta língua.

No que diz respeito à estrutura dos *prompts*, observou-se uma evolução progressiva ao longo do trabalho, possível de identificar na Tabela 8. Inicialmente, foram utilizados *prompts* únicos, contendo toda a informação relevante num só texto. Contudo, verificou-se que esta abordagem resultava frequentemente em respostas incompletas ou ambíguas, principalmente para problemas mais complexos. Com base nestas observações, adotou-se a divisão da informação em vários *prompts* consecutivos (estratégia de *prompt chaining*). Esta abordagem permitiu dividir a tarefa em múltiplas etapas, facilitando a compreensão progressiva por parte dos modelos. Mais tarde, aperfeiçoou-se novamente os *prompts*, passando a organizar a informação em pontos separados, com uma estrutura clara e sequencial. Esta forma de apresentação demonstrou ser particularmente eficaz na redução de ambiguidade, orientando melhor a resposta e garantindo maior coesão no raciocínio do modelo. De forma a ajudar utilizadores dos LLMs a identificar aspetos a aplicar nos *prompts*, foi desenvolvido um manual de *prompting* para problemas de Escalonamento das Operações, apresentado no Apêndice A.

Tabela 8 - Evolução dos *prompts*

Exemplo de *prompt* inicial

I want to solve an operation scheduling problem, specifically a Job Shop scheduling problem, with the objective of minimizing makespan.

This problem presents 3 jobs and 3 machines, where any job can be processed in any machine. Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine. There are some restrictions. First, each machine can only process one operation at a time. Every machine is available at the start. There are no precedence constraints between operations from different machines.

Here are the processing times (in minutes): Job 1: $t_{1,1} = 7$ (in machine 1); $t_{2,1} = 6$ (in machine 2); $t_{3,1} = 6$ (in machine 3). Job 2: $t_{1,2} = 6$ (in machine 2); $t_{2,2} = 7$ (in machine 3); $t_{3,2} = 7$ (in machine 1). Job 3: $t_{1,3} = 8$ (in machine 3); $t_{2,3} = 5$ (in machine 2); $t_{3,3} = 9$ (in machine 1)

Exemplo de *prompts* finais

1º *prompt*

I have a Job Shop Scheduling Problem for you to solve. I will present the details of this problem during the next prompts and after that I will ask you to give me a solution.

2º *prompt*

First, here are the processing times of the operations of every job, as well as the machines required to process them, in order of the pre-determined sequence.

J1: (M0, 7) -> (M1, 6) -> (M2, 6).

J2: (M1, 6) -> (M2, 7) -> (M0, 7).

J3: (M2, 8) -> (M1, 5) -> (M0, 9).

Remember these times and sequences to apply to this problem

3º *prompt*

The objective of this Scheduling Problem is minimizing makespan. This problem presents 3 jobs and 3 machines, where any job can be processed in any machine.

There are some restrictions that must be respected at all times:

- 1. Operation Sequence: Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine.*
- 2. Machine Overlap: Each machine can only process one operation at a time.*
- 3. Machine Availability: Every machine is available at the start.*
- 4. Precedence Constraints: An operation can only start processing after the previous operation from the same job finishes processing. There are no precedence constraints between operations from different machines.*

Using the method that is most appropriate for this case, give me an optimal solution to reduce total makespan (without giving me code to run on my PC). Focus on giving accurate information and complying with all restrictions, even if it takes more time to process the answer.

Apesar das melhorias introduzidas na formulação dos *prompts*, verificou-se que nem sempre as respostas eram corretas. Foram identificados diversos erros de raciocínio, por vezes bastante evidentes, que comprometiam a precisão das soluções apresentadas. Os erros em questão incluíam a sobreposição de operações na mesma máquina, o processamento de operações da mesma tarefa simultaneamente, ou a violação da sequência de execução de operações da mesma tarefa. Em certos casos, o LLM apresentava um determinado *makespan*, seguido de tempos de início de processamento das operações que não correspondiam a esse mesmo *makespan*.

Devido a esta instabilidade dos resultados, tornou-se essencial verificar manualmente todas as tabelas geradas, comparando os tempos de processamento, as máquinas envolvidas e as relações de precedência entre operações. Durante um teste de aplicação dos LLMs para otimizar uma instância de 5Jx3M – 5 tarefas em 3 máquinas, o modelo GPT o4-mini obteve um *makespan* total ótimo de 32 unidades de tempo. À primeira vista aparentava respeitar todas as restrições, no entanto, com uma inspeção mais pormenorizada do gráfico de Gantt apresentado na Figura 23, revelou que existia sobreposição de duas operações da mesma tarefa J5. Após um segundo pedido ao mesmo modelo para resolver este problema, obteve-se outra solução ótima com um *makespan* de 32 unidades de tempo, no entanto, como pode ser visto no gráfico de Gantt da Figura 24, apesar da semelhança com a primeira solução, desta vez não existe nenhuma sobreposição de operações. Este processo de validação foi fundamental para garantir que as soluções propostas eram viáveis. Observou-se que ao longo do tempo os modelos começaram a apresentar uma maior consistência, adaptando-se à metodologia e à estrutura dos prompts implementadas.

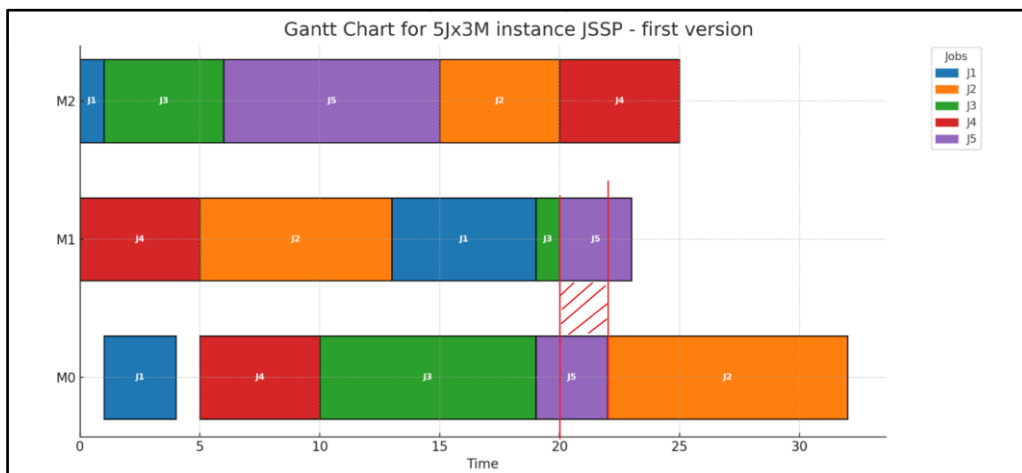


Figura 23 - Gráfico de Gantt para a primeira tentativa de solução de uma instância 5Jx3M

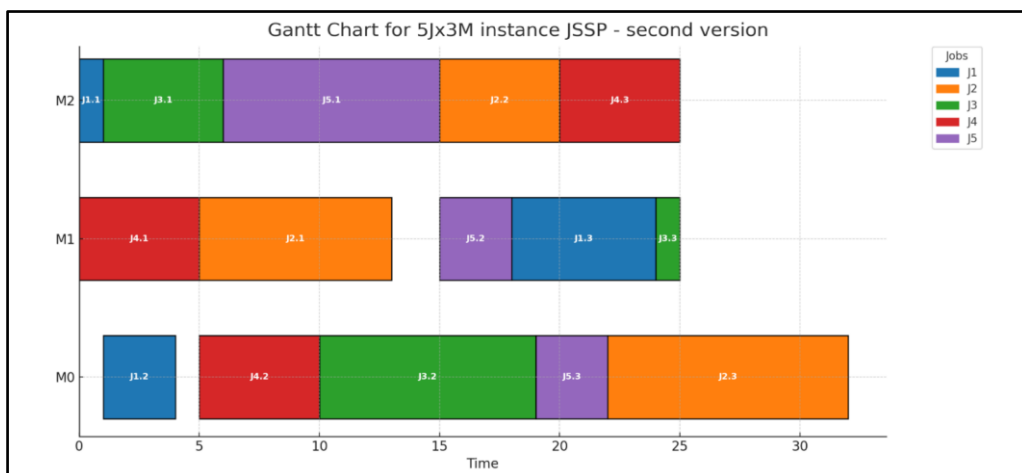


Figura 24 - Gráfico de Gantt para a segunda tentativa de solução de uma instância 5Jx3M

Para além da estrutura, outra decisão fundamental consistiu na forma como era feita a solicitação da resolução do problema. Foi especificado que não deveria ser utilizado código diretamente ou ferramentas externas, como por exemplo as OR-Tools. Esta indicação surgiu após se verificar que os modelos, por padrão, tentavam inicialmente resolver os problemas através da execução de código que não era diretamente aplicável, levando sempre a um impasse que atrasava o processo e introduzia redundâncias. Em vez disso, foi solicitado aos modelos que aplicassem o método mais adequado a cada problema, explicando o seu raciocínio passo a passo, sem recorrer a implementações computacionais. Este procedimento não só evitou passos desnecessários, como a tentativa de acesso a funções indisponíveis, como também permitiu uma análise mais direta da lógica adotada pelos modelos.

Com este conjunto de medidas, criou-se uma base metodológica robusta que permitiu avaliar com maior rigor o desempenho dos LLMs na resolução de problemas de Escalonamento. Esta preparação revelou-se essencial para garantir a reprodução dos resultados, bem como para estabelecer uma linha de raciocínio clara e objetiva ao longo de todo o processo experimental.

3.4. Aplicação dos LLMs na resolução de problemas de Escalonamento

Nesta fase do trabalho já foi adquirido conhecimento sobre os problemas de Escalonamento das Operações, em específico os problemas em ambiente de *Job Shop*, e sobre o funcionamento dos LLMs a serem avaliados, o ChatGPT (com o modelo GPT o4-mini) e o DeepSeek (com o modelo DeepSeek R1). Assim, foi considerado relevante questionar inicialmente os modelos sobre o seu conhecimento sobre o tópico, antes de pedir para resolver instâncias de *Job Shop*.

Para tal foi colocado o mesmo conjunto de perguntas a ambas as plataformas, em novas interações, de forma a evitar a influência de qualquer informação prévia. Foi adotada uma abordagem “*top-down*” para os *prompts*, isto é, começando por extrair a informação mais abrangente primeiro e depois concentrando o foco em colocar questões mais específicas. Durante esta fase de averiguação serão avaliadas as respostas fornecidas de um ponto de vista geral e de um ponto de vista técnico.

O primeiro *prompt* e as respostas de ambos os modelos estão apresentados na Tabela 9.

Tabela 9 - 1º *prompt*: conhecimento base sobre Escalonamento das Operações

Prompt 1: Do you know what the operation scheduling problem is?	
ChatGPT	<p>Yes! The Operation Scheduling Problem refers to optimizing the allocation and sequencing of operations in a manufacturing or service environment to achieve objectives like minimizing completion time, reducing costs, or maximizing efficiency.</p> <p><u>Key Aspects of the Problem:</u></p> <p>Operations & Tasks: Jobs consist of multiple operations that need to be scheduled.</p> <p>Resources: Machines, workers, or other resources are limited and must be allocated efficiently.</p>

Prompt 1: Do you know what the operation scheduling problem is?**Constraints:**

Precedence constraints (some operations must finish before others start).

Machine constraints (each machine can process only one task at a time).

Setup times (switching between tasks may require time).

Objective Functions:

Minimize makespan (total time to complete all tasks).

Minimize lateness/tardiness (how much tasks exceed deadlines).

Maximize resource utilization.

Common Types:

Job Shop Scheduling – Each job has a unique sequence of operations on different machines.

Flow Shop Scheduling – Jobs go through machines in the same order.

Open Shop Scheduling – No predefined order of operations.

Single Machine & Parallel Machine Scheduling – Jobs are assigned to one or multiple machines.

DeepSeek

Yes! The Operation Scheduling Problem is a well-known optimization challenge in operations research, manufacturing, logistics, and computer systems. It involves assigning tasks (operations) to resources (machines, workers, processors, etc.) over time to optimize certain objectives while satisfying constraints.

Key Components of the Problem:

Tasks/Jobs – Units of work that need to be processed.

Resources/Machines – Entities that perform the tasks (single or multiple).

Constraints – Such as precedence relations, deadlines, setup times, etc.

Objective Function – What needs to be optimized (e.g., makespan, tardiness, cost).

Common Variants:

Job Shop Scheduling (JSP): Jobs follow predefined machine sequences.

Flow Shop Scheduling: All jobs follow the same machine sequence.

Open Shop Scheduling: Jobs have no fixed machine order.

Parallel Machine Scheduling: Jobs are assigned to multiple identical or unrelated machines.

Project Scheduling (RCPS): Tasks have resource constraints in addition to precedence.

Objectives:

Minimize makespan (total completion time).

Minimize total tardiness (meeting deadlines).

Prompt 1: Do you know what the operation scheduling problem is?	
	<p><i>Maximize resource utilization.</i></p> <p><i>Minimize total cost.</i></p> <p>Complexity:</p> <p><i>Most scheduling problems are NP-Hard, meaning exact solutions are computationally expensive for large instances. Hence, heuristics (e.g., genetic algorithms, simulated annealing) and approximation methods are often used.</i></p>

Este *prompt* foi utilizado para analisar o conhecimento de ambas as plataformas sobre o tema mais abrangente: o Escalonamento das Operações. Sem qualquer troca de informação prévia, tanto o ChatGPT como o DeepSeek demonstraram um domínio técnico sobre este tópico, apresentando os elementos principais dos problemas de Escalonamento, tais como as tarefas e operações, as máquinas, as restrições e os objetivos. Para além disso, ambas as plataformas apresentaram diferentes ambientes de produção, como *Job Shop* e *Flow Shop*, e possíveis objetivos para os problemas, como a redução de *makespan* ou da utilização total dos recursos. A principal diferença entre o ChatGPT e o DeepSeek encontra-se na concisão da informação apresentada. O DeepSeek utiliza uma escrita mais detalhada do que o ChatGPT, abordando um tópico mais específico, a complexidade dos problemas de Escalonamento e a necessidade de aplicação de métodos exatos para obter uma solução ótima.

Com a noção de que ambos modelos compreendem a base dos problemas de Escalonamento das Operações, procurou-se saber a extensão do conhecimento que possuem em específico sobre problemas em ambiente de *Job Shop*. A Tabela 10 mostra as respostas de ambas as plataformas quando pedida uma breve descrição deste problema.

Tabela 10 - 2ª *prompt*: conhecimento sobre JSSP

Prompt 2: Explain the Job Shop Scheduling Problem in a concise manner, as if you were an expert in industrial management.	
ChatGPT	<p><i>The Job Shop Scheduling Problem (JSSP) asks: Given a set of jobs—each a fixed sequence of operations—and a set of machines, how do you assign and order every operation so that no machine handles more than one task at once, all precedence constraints are met, and a chosen metric (usually makespan) is minimized?</i></p> <p>Jobs & Operations: <i>Each job has ordered steps, each requiring a specific machine.</i></p> <p>Constraints: <i>Operations in a job must follow sequence; machines process one operation at a time.</i></p> <p>Objective: <i>Typically minimize the time to complete all jobs (makespan) or total lateness.</i></p> <p>Complexity: <i>NP-hard—exact methods work only for small instances.</i></p> <p>Practical Solutions: <i>Priority dispatch rules for speed, metaheuristics (GA, Tabu) for better quality, and hybrid real-time systems for adaptability.</i></p>

Prompt 2: Explain the Job Shop Scheduling Problem in a concise manner, as if you were an expert in industrial management.

DeepSeek	<p>Core Challenge: Schedule multiple jobs (each with a unique sequence of operations) on shared machines.</p> <p>Constraints:</p> <p>Each operation requires a specific machine for a fixed duration.</p> <p>A machine can only process one job at a time.</p> <p>Operations cannot be interrupted.</p> <p>Primary Objective: Minimize Makespan (the total time to complete all jobs).</p> <p>Industrial Goal: Maximize resource utilization, reduce work-in-process inventory, and meet delivery deadlines efficiently.</p> <p>Complexity: NP-hard – optimal solutions are computationally infeasible for real-world scale; requires heuristics (e.g., dispatching rules, genetic algorithms) or specialized scheduling software.</p> <p>In essence: Optimally sequence complex, customized production flows through constrained resources to minimize total completion time.</p>
-----------------	--

Este segundo *prompt* teve como objetivo avaliar a compreensão dos modelos sobre o problema de Escalonamento das Operações em *Job Shop* (JSSP). Tanto o ChatGPT como o DeepSeek apresentam definições corretas do problema, abordando as suas principais características: conjunto de tarefas que possuem uma sequência fixa de operações, as máquinas apenas poderem processar uma operação de cada vez e o objetivo mais comum ser a minimização do *makespan*.

É possível, no entanto, identificar uma diferença sutil no estilo de escrita de cada modelo, com o DeepSeek a focar-se num contexto industrial real, fazendo referência a objetivos operacionais concretos para além da redução do *makespan*, como a redução de inventário ou o cumprimento de prazos de entrega, e salientando a dificuldade de otimizar problemas de grande dimensão, enquanto que o ChatGPT ofereceu uma resposta mais genérica.

Por último, procurou identificar quais os métodos que ambas as plataformas sugeriam aplicar para solucionar um problema JSSP. As respostas estão apresentadas na Tabela 11.

Tabela 11 - 3º *prompt*: conhecimento sobre métodos de otimização

Prompt 3: What methods could I use to solve a Job Shop Scheduling Problem?	
ChatGPT	<p>To solve a Job Shop Scheduling Problem (JSSP), you can use one or more of the following method categories, depending on the problem size, solution quality needs, and time constraints:</p> <p>◆ <u>1. Exact Methods (Optimal, but slow for large problems)</u></p> <p>Mixed-Integer Linear Programming (MILP)</p>

Prompt 3: What methods could I use to solve a Job Shop Scheduling Problem?

Formulate constraints and objective as linear equations.

Solved with solvers like CPLEX or Gurobi.

Best for small instances (e.g. <15 jobs/machines).

Constraint Programming (CP)

Defines the problem via logical constraints (precedence, resource use).

Good for combinatorial constraints, flexible formulations.

Solvers: Google OR-Tools, IBM CP Optimizer.

Branch and Bound / Branch and Cut

Systematically explores all solutions, pruning suboptimal paths.

Computationally expensive.

◆ 2. Heuristics (Fast, but not always optimal)

Dispatching Rules

Examples: Shortest Processing Time (SPT), Earliest Due Date (EDD), Most Work Remaining (MWR).

Very fast and easy to implement, used in dynamic or real-time settings.

Typically suboptimal.

Shifting Bottleneck Heuristic

Prioritizes and schedules the most critical machines first.

High-quality results, especially for classical JSSPs.

◆ 3. Metaheuristics (Good balance of quality and time)

Genetic Algorithms (GA)

Evolves a population of schedules over generations.

Good for global exploration.

Simulated Annealing (SA)

Probabilistic method allowing occasional uphill moves to escape local optima.

Tabu Search

Explores neighbors while avoiding cycles using memory (tabu list).

Effective for many scheduling problems.

Ant Colony Optimization, Particle Swarm Optimization

Inspired by natural systems; effective for complex variants.

◆ 4. Hybrid Methods

Combine metaheuristics + exact methods for better performance.

Prompt 3: What methods could I use to solve a Job Shop Scheduling Problem?

Example: Use MILP to optimize machine-level subproblems inside a GA framework.

Useful for large-scale, real-world applications.

◆ 5. Reinforcement Learning / Machine Learning (Emerging Area)

Learn policies for dispatching or sequencing from simulation or historical data.

Promising for highly dynamic environments or when rules are hard to handcraft.

DeepSeek

Methods to Solve the Job Shop Scheduling Problem (JSSP)

1. Exact Methods (Guarantee optimality, but only for small problems):

Use **Mathematical Programming** (like Mixed-Integer Linear Programming - MIP or Constraint Programming - CP) or **Branch and Bound** algorithms. These are solved with commercial solvers like CPLEX or Gurobi. They are practical only for very small instances (e.g., under 10 jobs/machines) where absolute optimality is required.

2. Heuristics (Fast, simple rules for real-time scheduling):

Apply dispatching rules to prioritize jobs dynamically at machines. Common rules include:

SPT (Shortest Processing Time): Selects the job with the quickest next operation. Minimizes average flow time.

EDD (Earliest Due Date): Prioritizes jobs with the closest deadlines. Improves on-time delivery.

MWKR (Most Work Remaining): Focuses on jobs with the most total processing time left. Helps bottleneck resources.

FIFO (First-In-First-Out): Processes jobs in arrival order. Simple and fair.

These are ideal for dynamic shop floors, daily scheduling, or embedded in MES/SCADA systems.

3. Metaheuristics (Near-optimal solutions for real-world scale):

These explore the solution space efficiently for larger problems (10–100+ jobs/machines):

Local Search Methods:

Tabu Search: Uses a "tabu list" to avoid revisiting recent solutions, escaping local optima.

Simulated Annealing: Accepts worse solutions probabilistically (based on a "temperature" parameter) to explore widely.

Population-Based Methods:

Genetic Algorithms (GA): Evolves a population of schedules using selection, crossover, and mutation operators.

Particle Swarm Optimization (PSO): Mimics social behavior to guide solutions toward better regions.

Problem-Specific Methods:

Prompt 3: What methods could I use to solve a Job Shop Scheduling Problem?

Shifting Bottleneck: Focuses sequentially on scheduling the most critical (bottleneck) machines first.

These are used in tactical planning (e.g., weekly/monthly schedules) and power Advanced Planning & Scheduling (APS) software.

4. Hybrid & Advanced Approaches:

Combine methods (e.g., metaheuristics with dispatching rules or MIP). Emerging techniques include:

Reinforcement Learning (RL): Trains AI agents to make scheduling decisions through trial and error.

Multi-Agent Systems: Autonomous negotiation between machines/jobs to create schedules.

No que diz respeito às respostas sobre os métodos para resolver o JSSP, ambas as plataformas demonstraram muito conhecimento, dividindo corretamente as abordagens em métodos exatos, heurísticas, meta-heurísticas e técnicas híbridas. De forma geral, ambos os LLMs fizeram referência a algoritmos mais tradicionais como *Branch and Bound*, heurísticas como *Genetic Algorithms* e estratégias emergentes como *Reinforcement Learning*.

Apesar de abordarem os mesmos conceitos, observaram-se algumas diferenças na ênfase dada a certos elementos. O ChatGPT apresentou com maior evidência os métodos exatos e incluiu referências a ferramentas de resolução como o OR-Tools e o IBM CP Optimizer. Já o DeepSeek destacou-se pela descrição mais detalhada das regras de prioridade e pela sua contextualização prática, associando os métodos a sistemas de planeamento industrial.

Através desta análise sucinta foi possível confirmar que ambos os LLMs possuem um elevado nível de conhecimento sobre o tema de Escalonamento das Operações, em particular nos problemas de *Job Shop*. Considerou-se relevante confirmar este conhecimento inicialmente, para certificar a interpretação correta por parte dos LLMs, dos problemas de Escalonamento das Operações a serem aplicados.

Passou-se de seguida para a fase de teste dos LLMs numa série de instâncias de Escalonamento das Operações em *Job Shop*. É necessário referir que foi preciso adaptar algumas instâncias frequentemente estudadas, tais como as apresentadas por *Fisher e Thompson*, por *Lawrence* e por *Storer, Wu e Vaccari* [107], modificando-as parcialmente de forma a evitar que os LLMs procurem resolver os problemas através de uma pesquisa da solução ótima na Internet.

O primeiro problema apresentado aos modelos GPT o4-mini e DeepSeek R1 consistiu em programar a ordem de processamento de 3 tarefas, cada uma composta por 3 operações com um percurso pré-determinado em 3 máquinas, com o objetivo de minimizar o *makespan* total. Considera-se que este problema, designado [3Jx3M], apresenta uma baixa complexidade derivada do seu número reduzido de tarefas e máquinas, sendo que em alguns problemas desta

dimensão é possível uma resolução mental, sem recorrer a nenhum método de otimização estruturado.

Na Tabela 12 e na Tabela 13 são apresentadas as interações realizadas com o GPT o4-mini e o DeepSeek R1 no contexto da resolução do problema [3Jx3M]. Os *inputs* para ambos os modelos são semelhantes, no entanto foi necessário adicionar o seguinte requisito ao DeepSeek, devido à observação prévia de várias ocorrências nas quais este modelo procurou resolver o problema precocemente, antes de adquirir a informação completa do problema – “*Only give me the final answer after I ask for it*”. Assim, certificou-se que o ChatGPT e o DeepSeek recebem os mesmos dados de entrada e apenas apresentam a solução após ser fornecido o último *prompt*.

Tabela 12 - Resolução do problema de Job Shop [3Jx3M] pelo GPT o4-mini

GPT o4-mini
<p><i>Prompt 1: “I have a Job Shop Scheduling Problem for you to solve. I will present the details of this problem during the next prompts and after that I will ask you to give me a solution.”</i></p>
<p><i>Great! I’m ready to help. When you’re ready, please provide:</i></p> <ul style="list-style-type: none"> • <i>The set of jobs and their sequences of operations</i> • <i>The machines each operation must run on</i> • <i>Processing times for each operation</i> • <i>Any release dates, due dates, or deadlines</i> • <i>Objective (e.g. minimize makespan, total tardiness, etc.)</i> <p><i>Once I have those details, I’ll work out a scheduling solution for you.</i></p>
<p><i>Prompt 2: “First, here are the processing times of the operations of every job, as well as the machines required to process them, in order of the pre-determined sequence. J1: (M0, 7) -> (M1, 6) -> (M2, 6). J2: (M1, 6) -> (M2, 7) -> (M0, 7). J3: (M2, 8) -> (M1, 5) -> (M0, 9). Remember these times and sequences to apply to this problem after the final details.”</i></p>
<p><i>Got it. Here’s what I’ve noted:</i></p> <ul style="list-style-type: none"> • <i>Job J1:</i> <ol style="list-style-type: none"> <i>1. Operation 1 on M0, duration 7</i> <i>2. Operation 2 on M1, duration 6</i> <i>3. Operation 3 on M2, duration 6</i> • <i>Job J2:</i> <ol style="list-style-type: none"> <i>1. Operation 1 on M1, duration 6</i> <i>2. Operation 2 on M2, duration 7</i> <i>3. Operation 3 on M0, duration 7</i> • <i>Job J3:</i> <ol style="list-style-type: none"> <i>1. Operation 1 on M2, duration 8</i> <i>2. Operation 2 on M1, duration 5</i> <i>3. Operation 3 on M0, duration 9</i> <p><i>I’ll hold onto these sequences and processing times. Please go ahead and share any release dates, due dates, deadlines, and the scheduling objective.</i></p>

GPT o4-mini

Prompt 3: "The objective of this Scheduling Problem is minimizing makespan. This problem presents 3 jobs and 3 machines, where any job can be processed in any machine. There are some restrictions that must be respected at all times: 1. Operation Sequence: Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine. 2. Machine Overlap: Each machine can only process one operation at a time. 3. Machine Availability: Every machine is available at the start. 4. Precedence Constraints: An operation can only start processing after the previous operation from the same job finishes processing. There are no precedence constraints between operations from different machines. Using the method that is most appropriate for this case, give me an optimal solution to reduce total makespan (without giving me code to run on my PC). Focus on giving accurate information and complying with all restrictions, even if it takes more time to process the answer."

Below is an optimal schedule achieving a makespan of 29 time-units. All precedence and machine-capacity constraints are respected.

Operation Start/Finish Times

Job Operation Machine Start Finish

J1	O ₁	M0	0	7
J1	O ₂	M1	13	19
J1	O ₃	M2	19	25
J2	O ₁	M1	0	6
J2	O ₂	M2	8	15
J2	O ₃	M0	22	29
J3	O ₁	M2	0	8
J3	O ₂	M1	8	13
J3	O ₃	M0	13	22

Explanation of Key Idle Periods

- *Machine M1 is idle from time 6 to 8 because J3's O₂ cannot start before its predecessor (J3-O₁) finishes at 8.*
- *Machine M2 is idle from time 15 to 19 awaiting J1's O₂ completion (at 19) so that J1-O₃ can begin.*
- *Machine M0 is idle from time 7 to 13 awaiting J3-O₂ (finishing at 13) before J3-O₃.*

The final completion is J2-O₃ at time 29, so the minimum makespan is 29. This schedule is optimal for the given 3×3 job-shop instance.

Os primeiros dois *prompts* têm o objetivo de fazer o ChatGPT focar totalmente o raciocínio na resolução de problemas de Escalonamento em *Job Shop* e de assimilar a informação da instância do problema gradualmente, antes de proceder à procura da solução ótima.

A resposta ao terceiro *prompt* foi gerada em 35 segundos, e apresenta um *makespan* de 29 u.t., a solução ótima para esta instância. Após inspeção da tabela indicativa dos tempos de início e fim de processamento das operações, verificou-se que não existe violação de nenhuma restrição, ou seja, a sequência de processamento das operações pertencentes à mesma tarefa é sempre respeitada e as máquinas apenas processam uma operação de cada vez, sendo assim considerado um programa viável. O ChatGPT apresentou ainda uma breve explicação dos tempos de inatividade das máquinas, resultantes em determinadas operações aguardarem pela conclusão de outras, como medida necessária a fim de evitar sobreposição de operações ou sequências de processamento alteradas.

De seguida colocou-se a questão de que métodos o modelo utilizou para alcançar esta solução e, aliado à informação proveniente da fase de pensamento para a resposta ao *prompt* 3, determinou-se o seguinte procedimento:

- Modelação do problema num grafo disjuntivo
- Cálculo do limite inferior LI (determinado pela máquina com maior carga C_m)
 - $\max(LI_1, LI_2) = \max(22, 23) = 23$ (8)
 - $LI_1 = \max(\sum_{i=1}^3 p_{i,1}, \sum_{i=1}^3 p_{i,2}, \sum_{i=1}^3 p_{i,3})$ (9)
 - $LI_2 = \max(C_1, C_2, C_3)$ (10)
- Pesquisa *Branch and Bound* para determinar a ordem das operações em cada máquina, auxiliada da regra *Earliest-Start*

Tabela 13 - Resolução do problema de Job Shop [3Jx3M] pelo DeepSeek R1

DeepSeek R1
<p><i>Prompt 1: "I have a Job Shop Scheduling Problem for you to solve. I will present the details of this problem during the next prompts and after that I will ask you to give me a solution. Only give me the final answer after I ask for it"</i></p>
<p><i>I understand. I'm ready to receive the details of your Job Shop Scheduling Problem. Please provide the following information in your next prompts:</i></p> <ul style="list-style-type: none"> • <i>Number of jobs</i> • <i>Number of machines</i> • <i>Operations for each job (in sequence), including:</i> <ul style="list-style-type: none"> ○ <i>Machine assigned for each operation</i> ○ <i>Processing time for each operation</i> <p><i>Once you provide all details, I'll compute the solution and wait for your instruction to share it.</i></p>
<p><i>Prompt 2: "First, here are the processing times of the operations of every job, as well as the machines required to process them, in order of the pre-determined sequence. J1: (M0, 7) -> (M1, 6) -> (M2, 6). J2: (M1, 6) -> (M2, 7) -> (M0, 7). J3: (M2, 8) -> (M1, 5) -> (M0, 9). Remember these times and sequences to apply to this problem after the final details."</i></p>

DeepSeek R1
<p><i>I have recorded the processing sequences for all jobs as follows:</i></p> <ul style="list-style-type: none"> • <i>J1: (M0, 7) → (M1, 6) → (M2, 6)</i> • <i>J2: (M1, 6) → (M2, 7) → (M0, 7)</i> • <i>J3: (M2, 8) → (M1, 5) → (M0, 9)</i> <p><i>I understand that this is part of the problem setup, and I will wait for any additional details in your next prompts. Once you provide all necessary information and explicitly ask for the solution, I will compute and deliver it. Please continue with any further details when ready.</i></p>
<p><i>Prompt 3: "The objective of this Scheduling Problem is minimizing makespan. This problem presents 3 jobs and 3 machines, where any job can be processed in any machine. There are some restrictions that must be respected at all times: 1. Operation Sequence: Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine. 2. Machine Overlap: Each machine can only process one operation at a time. 3. Machine Availability: Every machine is available at the start. 4. Precedence Constraints: An operation can only start processing after the previous operation from the same job finishes processing. There are no precedence constraints between operations from different machines. Using the method that is most appropriate for this case, give me an optimal solution to reduce total makespan (without giving me code to run on my PC). Focus on giving accurate information and complying with all restrictions, even if it takes more time to process the answer."</i></p>
<p><i>Optimal Solution for Job Shop Scheduling Problem</i></p> <p><i>Objective: Minimize makespan (total completion time)</i></p> <p><i>Makespan Achieved: 29 time units</i></p> <p><i>Schedule Details:</i></p> <p><i>Each operation is represented as:</i></p> <p><i>Job (Start Time, End Time) on a machine.</i></p> <ul style="list-style-type: none"> • <i>Machine M0:</i> <ul style="list-style-type: none"> ○ <i>J1: (0, 7)</i> ○ <i>J3: (13, 22)</i> ○ <i>J2: (22, 29)</i> • <i>Machine M1:</i> <ul style="list-style-type: none"> ○ <i>J2: (0, 6)</i> ○ <i>J3: (8, 13)</i> ○ <i>J1: (13, 19)</i> • <i>Machine M2:</i> <ul style="list-style-type: none"> ○ <i>J3: (0, 8)</i> ○ <i>J2: (8, 15)</i> ○ <i>J1: (19, 25)</i> <p><i>Job-Wise Sequence:</i></p> <ul style="list-style-type: none"> • <i>Job J1 (Completion: 25):</i> <ul style="list-style-type: none"> ○ <i>M0: (0, 7)</i> ○ <i>M1: (13, 19)</i>

DeepSeek R1
<ul style="list-style-type: none"> ○ <i>M2: (19, 25)</i> • <i>Job J2 (Completion: 29):</i> <ul style="list-style-type: none"> ○ <i>M1: (0, 6)</i> ○ <i>M2: (8, 15)</i> ○ <i>M0: (22, 29)</i> • <i>Job J3 (Completion: 22):</i> <ul style="list-style-type: none"> ○ <i>M2: (0, 8)</i> ○ <i>M1: (8, 13)</i> ○ <i>M0: (13, 22)</i> <p><i>Verification of Constraints:</i></p> <ol style="list-style-type: none"> 1. <i>Operation Sequence:</i> <ul style="list-style-type: none"> ○ <i>J1: M0 → M1 → M2 (as required: 0→7, 13→19, 19→25).</i> ○ <i>J2: M1 → M2 → M0 (as required: 0→6, 8→15, 22→29).</i> ○ <i>J3: M2 → M1 → M0 (as required: 0→8, 8→13, 13→22).</i> 2. <i>Machine Overlap:</i> <ul style="list-style-type: none"> ○ <i>Each machine processes one operation at a time:</i> <ul style="list-style-type: none"> ▪ <i>M0: J1 (0-7), then idle (7-13), then J3 (13-22), then J2 (22-29).</i> ▪ <i>M1: J2 (0-6), then idle (6-8), then J3 (8-13), then J1 (13-19).</i> ▪ <i>M2: J3 (0-8), then J2 (8-15), then idle (15-19), then J1 (19-25).</i> 3. <i>Machine Availability:</i> <ul style="list-style-type: none"> ○ <i>All machines start at time 0 (no delays).</i> 4. <i>Precedence Constraints:</i> <ul style="list-style-type: none"> ○ <i>For J1: M0 ends at 7 → M1 starts at 13 (7 ≤ 13). M1 ends at 19 → M2 starts at 19 (19 ≤ 19).</i> ○ <i>For J2: M1 ends at 6 → M2 starts at 8 (6 ≤ 8). M2 ends at 15 → M0 starts at 22 (15 ≤ 22).</i> ○ <i>For J3: M2 ends at 8 → M1 starts at 8 (8 ≤ 8). M1 ends at 13 → M0 starts at 13 (13 ≤ 13).</i>

Tal como foi mencionado anteriormente, para o DeepSeek foi reforçada no primeiro *prompt* a necessidade de esperar até receber a informação toda do problema antes de tentar encontrar a solução.

Como resposta ao terceiro *prompt* o DeepSeek gerou, em 131 segundos, uma solução que apresenta um *makespan* de 29 u.t, obtendo, tal como o ChatGPT, a solução ótima. Através dos tempos de início de processamento de cada operação, apresentados pelo modelo, confirmou-se que todas as restrições são respeitadas, tornando este calendário viável. Para além dos tempos de início de processamento das operações, apresenta também os tempos de

inatividade das máquinas (apesar de não conter o mesmo nível de detalhe que o ChatGPT) e a confirmação, passo-a-passo, do cumprimento das restrições de precedência para cada tarefa.

O método de resolução deste problema por parte do DeepSeek foi semelhante ao do ChatGPT, com a aplicação de um grafo disjuntivo inicialmente, o cálculo do limite inferior definido pela carga da máquina M_0 e a exploração sistemática de possíveis sequências nas máquinas (com um total de 216 combinações possíveis).

Este primeiro teste revelou que tanto o GPT o4-mini como o DeepSeek R1 não demonstraram dificuldades na resolução de um problema de baixa complexidade, sendo capazes de rapidamente encontrar a solução ótima do problema [3Jx3M], representada no gráfico de Gantt da Figura 25. Este resultado é explicado pela dimensão reduzida do problema testado, que limita significativamente o espaço de soluções possíveis, tornando viável a exploração exaustiva das soluções num intervalo de tempo limitado.

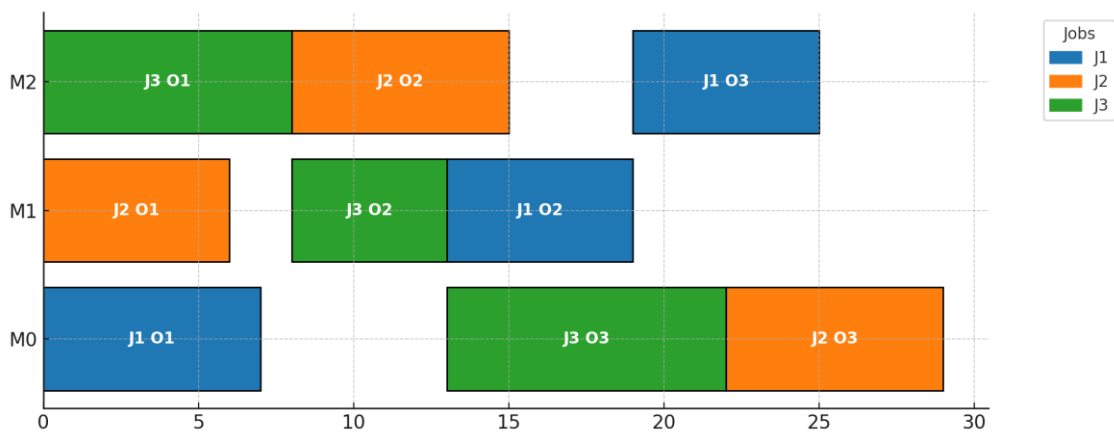


Figura 25 - Gráfico de Gantt representativo da solução ótima do problema [3Jx3M]

Com o objetivo de avaliar o desempenho em problemas de maior dimensão, e explorar os limites da capacidade de raciocínio dos modelos, optou-se por aumentar gradualmente o número de tarefas e o número de máquinas incluídas nas instâncias de teste, resultando em 19 instâncias distintas analisadas, listadas na Tabela 14.

Inicialmente o processo de aumentar a complexidade das instâncias consistiu em fixar o número de máquinas presentes nos problemas ($m=3$) e apenas aumentar o número de tarefas a serem atribuídas às três máquinas. Quando começou a verificar-se não só um aumento de dificuldade em encontrar o *makespan* ótimo, como também a violação sistemática das restrições impostas e a existência de vários erros de continuidade nos programas gerados (ex. certas operações apresentarem diferentes tempos de início de processamento, na representação do programa por máquina e na representação do programa por tarefa), reduziu-se significativamente o número de tarefas das novas instâncias, aumentando o número de máquinas para $m=4$. Deste modo, foram analisadas várias instâncias com três, quatro e cinco máquinas, para além de algumas instâncias mais complexas abordadas posteriormente.

Tabela 14 - Lista de instâncias usadas para teste nos LLMs

Instância	Tarefas	Máquinas	Nº total de Operações
[3Jx3M]	3	3	9
[5Jx3M]	5	3	15
[8Jx3M]	8	3	24
[10Jx3M]	10	3	30
[3Jx4M]	3	4	12
[5Jx4M]	5	4	20
[8Jx4M]	8	4	32
[10Jx4M]	10	4	40
[3Jx5M]	3	5	15
[5Jx5M]	5	5	25
[8Jx5M]	8	5	40
[10Jx5M]	10	5	50
[15Jx5M]	15	5	75
[20Jx5M]	20	5	100
[6Jx6M]	6	6	36
[7Jx7M]	7	7	49
[8Jx8M]	8	8	64
[9Jx9M]	9	9	81
[10Jx10M]	10	10	100

Considerando as instâncias apresentadas na Tabela 14, observa-se um aumento considerável na dimensão dos problemas em teste, desde a primeira instância [3Jx3M] que contempla 9 operações diferentes, até à instância [10Jx10M] com um total de 100 operações.

De seguida é apresentado o problema de *Job Shop* relativo ao processamento de 10 tarefas em 3 máquinas, [10Jx3M]. Apesar de inicialmente não parecer um incremento tão substancial da dimensão do problema relativamente à instância [3Jx3M], aumentando de 9 para 30 operações, estes problemas de Escalonamento mostram um crescimento exponencial da complexidade devido ao número muito elevado de possibilidades de calendarização das operações. Enquanto o problema anterior apresentava 216 possíveis ordens de calendarização das operações, este apresenta um número extremamente elevado de $10! \times 10! \times 10! = 4.788e^{19}$ possibilidades (10! ordens possíveis em cada uma das 3 máquinas). Para além disso com mais possibilidades

de ordenação das operações em cada máquina, é também criada uma maior chance de existir sobreposição de operações, o que conduz a uma maior variabilidade nos tempos de inatividade dos problemas.

Assim como no problema anterior, foram colocados os três *prompts* segundo o *template* estabelecido no manual, aos modelos GPT o4-mini e DeepSeek R1 para resolver o problema [10Jx3M]. O primeiro teste realizado no ChatGPT gerou, em 72 segundos, o programa de processamento das operações observado na Tabela 15 e no gráfico da Figura 26, com um *makespan* de 694 u.t.

Tabela 15 - Primeiro programa gerado pelo GPT o4-mini para [10Jx3M]

Tarefa	Operação	Máquina	Início	Fim	Duração
J7	1	M2	0	12	12
J10	1	M3	0	18	18
J1	1	M1	0	20	20
J2	1	M3	18	50	32
J1	2	M2	20	51	31
J5	1	M1	20	62	42
J1	3	M3	51	68	17
J10	2	M2	51	88	37
J2	2	M1	62	86	24
J5	2	M3	68	85	17
J4	1	M3	85	171	86
J2	3	M2	88	106	18
J10	3	M1	88	149	61
J5	3	M2	106	152	46
J9	1	M1	149	224	75
J8	1	M2	152	215	63
J6	1	M2	215	282	67
J9	2	M3	224	274	50
J7	2	M1	224	304	80
J9	3	M2	282	323	41
J7	3	M3	304	354	50
J8	2	M1	304	398	94
J3	1	M2	323	395	72
J3	2	M3	395	418	23
J4	2	M2	395	471	76
J3	3	M1	418	476	58
J8	3	M3	418	516	98
J4	3	M1	476	534	58
J6	2	M1	534	632	98
J6	3	M3	632	694	62

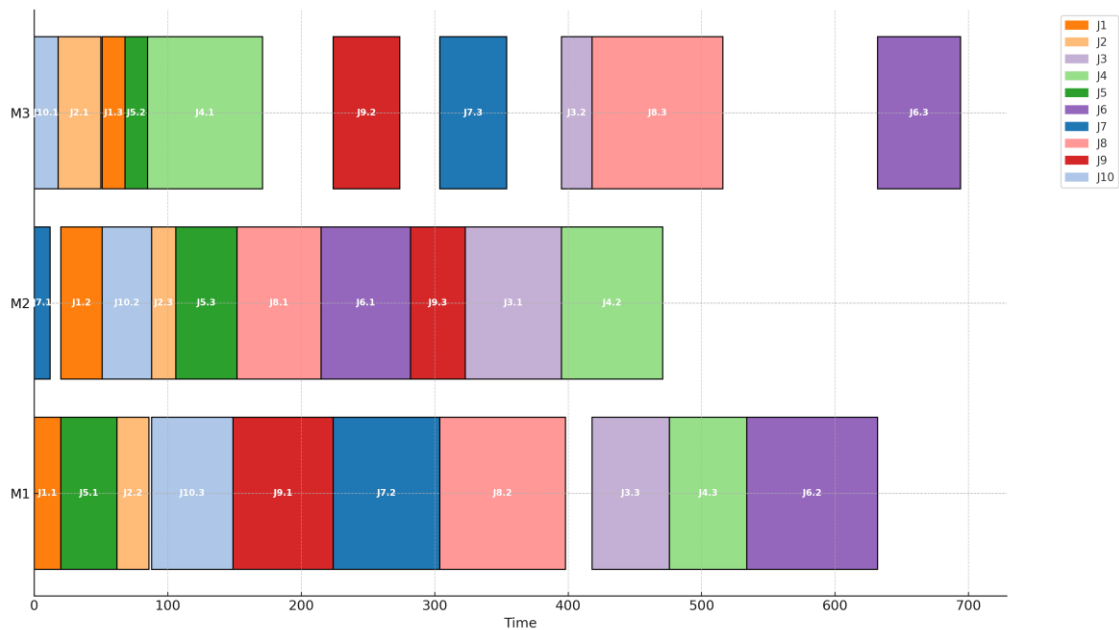


Figura 26 - Primeiro gráfico Gantt gerado pelo GPT o4-mini para [10Jx3M]

Esta solução foi encontrada pelo ChatGPT através do seguinte procedimento:

- **Regras de prioridade gulosas** – o modelo juntou duas regras de prioridade, *Earliest Start Time* (EST) e *Shortest Processing Time* (SPT), para escolher a operação a colocar em cada máquina. Em cada etapa deste processo, o modelo analisou uma determinada máquina m , e para cada operação disponível nesse momento calculou o tempo de conclusão mais cedo possível $EFT_{i,j} = EST_{i,j} + p_{i,j}$, escolhendo a operação com o menor $EFT_{i,j}$. Depois de tomada a decisão da operação a introduzir no programa, todas as outras operações que concorrem pela mesma máquina m tiveram o seu EST atualizado. Este processo é considerado guloso porque escolhe sempre a opção com o melhor valor sem consideração das implicações para operações futuras.
- **Cálculo do limite inferior LI** – para confirmar se esta solução é possível e ótima foram calculados os limites inferiores da carga das máquinas e do tempo de processamento total das tarefas. Considerando que o LI calculado é menor que o *makespan* encontrado ($610 < 694$), é possível que esta solução seja apenas aproximada do ótimo (no entanto nem sempre o *makespan* ótimo é igual ao limite inferior).

$$\circ \max(LI_1, LI_2) = \max(255, 610) = 610 \quad (11)$$

$$\square LI_1 = \max(\sum_{i=1}^3 p_{i,1}; \sum_{i=1}^3 p_{i,2}; (\dots); \sum_{i=1}^3 p_{i,10}) \quad (12)$$

$$\square LI_2 = \max(C_1, C_2, C_3) \quad (13)$$

Apesar desta solução ser possível, isto é, não viola nenhuma restrição imposta ao modelo, sabe-se que não é considerada uma solução ótima. Isto porque, através da formulação matemática deste problema em *Constraint Programming* no software *CPLEX Optimization Studio*, alcançou-se o *makespan* de 610 u.t., o limite inferior relativo à carga da máquina M_1 e valor mínimo considerando as restrições impostas e os parâmetros definidos inicialmente.

Com o objetivo de maximizar a qualidade da solução gerada, otimizando o *makespan*, realizou-se um segundo teste no ChatGPT utilizando os mesmos *prompts*. Durante este teste o modelo optou por uma estratégia de resolução semelhante, no entanto com diferenças consideráveis na ordem de aplicar os métodos de otimização, obtendo o *makespan* ótimo de 610 u.t. em 38 segundos, resultado da solução exposta na Tabela 16 e no gráfico de Gantt da Figura 27.

Tabela 16 - Segundo programa gerado pelo GPT o4-mini para [10x3M]

Tarefa	Operação	Máquina	Início	Fim	Duração
J1	1	M1	0	20	20
J7	1	M2	0	12	12
J10	1	M3	0	18	18
J8	1	M2	12	75	63
J2	1	M3	18	50	32
J5	1	M1	20	62	42
J4	1	M3	50	136	86
J2	2	M1	62	86	24
J1	2	M2	75	106	31
J9	1	M1	86	161	75
J2	3	M2	106	124	18
J10	2	M2	124	161	37
J1	3	M3	136	153	17
J5	2	M3	153	170	17
J10	3	M1	161	222	61
J6	1	M2	161	228	67
J9	2	M3	170	220	50
J7	2	M1	222	302	80
J9	3	M2	228	269	41
J5	3	M2	269	315	46
J8	2	M1	302	396	94
J7	3	M3	302	352	50
J3	1	M2	315	387	72
J4	2	M2	387	463	76
J3	2	M3	387	410	23
J6	2	M1	396	494	98
J8	3	M3	410	508	98
J3	3	M1	494	552	58
J6	3	M3	508	570	62
J4	3	M1	552	610	58

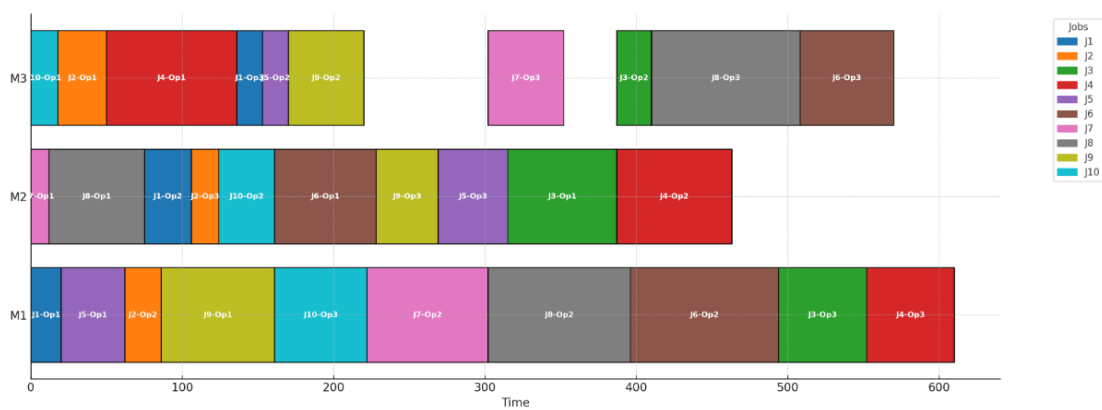


Figura 27 - Segundo gráfico Gantt gerado pelo GPT o4-mini para [10Jx3M]

Apesar das regras de prioridade assegurarem um calendário possível sem sobreposição de operações ou violações da ordem de precedência, estes métodos de otimização são determinísticos, resultando sempre em calendários iguais quando utilizadas as mesmas regras de prioridade na mesma ordem num problema de Escalonamento específico. É possível estes métodos originarem apenas um *makespan* aproximado, como no primeiro teste realizado para esta instância, o que torna a escolha do procedimento de resolução muito importante para encontrar a solução ótima. Para o modelo alcançar o *makespan* de 610 u.t. neste segundo teste, recorreu-se ao seguinte procedimento:

- **Gráfico disjuntivo** – o modelo simulou um gráfico disjuntivo, com os arcos conjuntivos a representar as restrições de precedência de operações pertencentes à mesma tarefa e os arcos disjuntivos a relacionar as operações que competem pela mesma máquina, como forma de assegurar que respeita estas restrições.
- **Aplicação do algoritmo de Giffler-Thompson** – esta heurística construtiva procurou, por etapas, todas as operações disponíveis no momento e calculou, para cada operação pertencente à máquina m , o tempo de início de processamento mais cedo possível (EST), ou seja, o valor máximo entre o tempo em que a máquina m fica disponível e o tempo em que a operação anterior da mesma tarefa termina o processamento. Depois, ao escolher e introduzir a operação com menor EST para a máquina m (operação que pode começar mais cedo), todas as outras operações que contestam essa máquina são adiadas e os seus tempos EST atualizados.
- **Em caso de desempate** – quando múltiplas operações contêm o mesmo EST , a escolha da operação a ser introduzida primeiro no programa foi feita através da heurística *Shortest Processing Time* (SPT), valorizando as operações com menor tempo de processamento.

A principal diferença entre estas duas soluções está na prioridade de aplicação das heurísticas utilizadas. Enquanto no primeiro caso o ChatGPT considerou o tempo de início de processamento mais cedo (EST) juntamente com o tempo de processamento da operação ($p_{i,j}$), na segunda tentativa foi considerado primeiro o EST e só depois, como forma de desempate, o SPT . Isto é evidente na escolha da segunda operação da máquina M_2 . No primeiro teste foi valorizado o tempo de conclusão da operação 2 da tarefa 1 ($EFT = 51$)

apesar de ser necessário adiar o seu começo ($EST = 20$). Já no segundo teste, como estabelecido, foram primeiro consideradas as operações com menor EST , sendo que a operação 1 das tarefas 3, 6 e 8 estavam disponíveis para iniciar logo quando fosse possível ($EST = 12$). Como a operação 1 da tarefa 8 possui o menor tempo de processamento entre estas opções ($SPT = 63$) foi introduzida primeiro, independentemente do seu tempo de conclusão ($EFT = 75$). Através destes dois testes é possível observar a relevância da exploração de vários métodos de otimização para maximizar a probabilidade de obter soluções ótimas em problemas de *Job Shop*, mesmo quando se recorrem a heurísticas semelhantes.

A mesma instância [10x3M] foi usada para testar o desempenho do modelo DeepSeek R1 na resolução de problemas desta escala, no entanto, este não apresentou resultados tão satisfatórios quanto o GPT o4-mini. Foram efetuadas três tentativas de encontrar o *makespan* ótimo para esta instância, culminando em três calendários de Escalonamento impossíveis. Na Figura 28 está representado o gráfico Gantt de uma das soluções geradas pelo DeepSeek, em 9 minutos e 47 segundos, com o objetivo de demonstrar os erros encontrados.

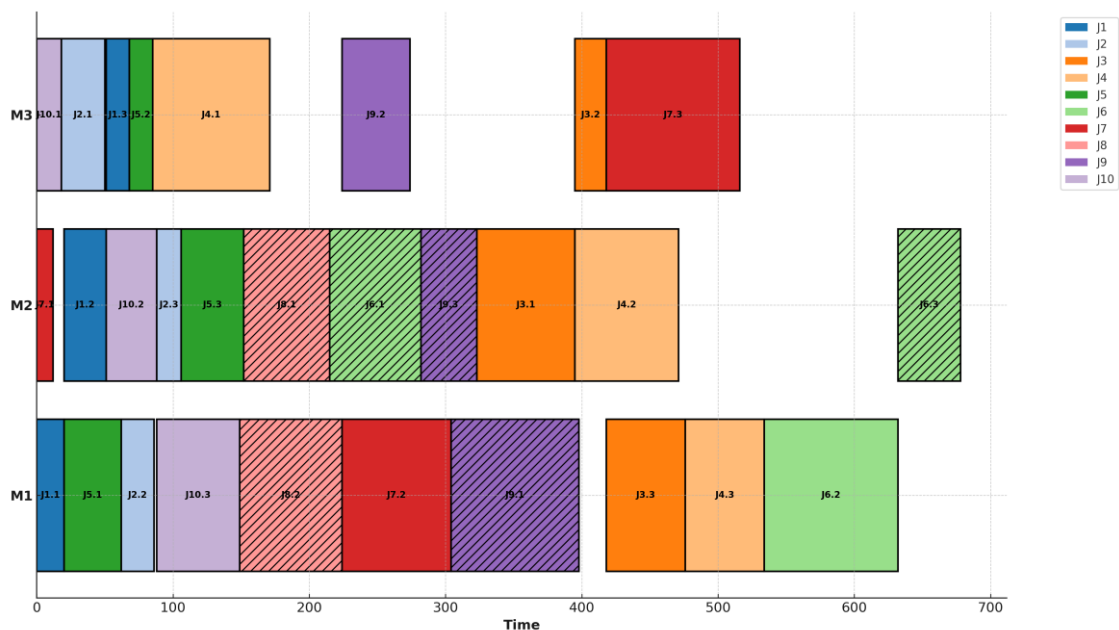


Figura 28 - Gráfico Gantt gerado pelo DeepSeek R1 para [10x3M]

Esta solução apresenta um *makespan* de 678 u.t., um valor aproximado do ótimo, no entanto observa-se no gráfico que várias restrições não foram respeitadas e que foram cometidos múltiplos erros:

- $J6.1$ e $J6.3$ estão representadas na mesma máquina (M_2), apesar de ser explícita a ordem de processamento e as máquinas atribuídas a cada operação desta tarefa ($J6.1 (M_2) \rightarrow J6.2 (M_1) \rightarrow J6.3 (M_3)$). Nesta situação, simplesmente mover $J6.3$ para a máquina M_3 , com o tempo de início de processamento igual, iria resolver o problema.
- $J8.1$ e $J8.2$ decorrem parcialmente ao mesmo tempo, infringindo a restrição de sobreposição de operações da mesma tarefa. Para além disso $J8.3$, atribuída a M_3 , não está representada neste programa. Sendo que as primeiras duas operações desta tarefa

estão colocadas a meio do ciclo de processamento das respetivas máquinas, uma alteração iria comprometer o seguimento do programa.

- Do mesmo modo, $J9$ compromete a restrição de sobreposição de operações da mesma tarefa, com $J9.1$ e $J9.3$ decorrendo parcialmente ao mesmo. A restrição de precedência também não é respeitada, com $J9.1$ iniciando o seu processamento depois de $J9.2$ estar completa e a meio do processamento de $J9.3$.

Com o aumento da dimensão dos problemas de teste, foram identificados gradualmente mais erros de natureza dos expostos pela resolução do problema [10Jx3M] pelo DeepSeek R1, em ambos os modelos a serem testados. Foram identificadas também infrações da restrição de sobreposição de operações na mesma máquina, incoerências entre o *makespan* final e o tempo de conclusão da última operação apresentado nas tabelas, e a definição de um *makespan* impossível (menor do que o limite inferior definido pela carga de uma máquina ou pelo tempo total de processamento de uma tarefa).

Face aos resultados insatisfatórios, foi tomada a decisão de testar também o modelo GPT o3 da versão ChatGPT Pro. Segundo a empresa OpenAI, este modelo lançado recentemente possui a melhor capacidade de raciocínio de todos os modelos disponíveis, sendo indicado para resolver problemas complexos. Como esperado, o GPT o3 não teve dificuldade em gerar soluções ótimas para os problemas de menor escala que já tinham sido testados nos outros dois modelos, demonstrando um desempenho elevado.

Da mesma forma, observou-se a eficácia do modelo GPT o3 quando testado com a instância que gerou complicações aos dois modelos anteriormente. Na primeira tentativa este modelo encontrou, em 4 minutos e 42 segundos, uma solução viável para o *makespan* ótimo de 610 u.t., representado na Figura 29 e na Tabela 17. Apesar de ter sido necessário esperar mais tempo na “fase de pensamento” deste modelo do que no GPT o4-mini, foi apresentada logo na primeira tentativa a solução ótima desta instância, sem qualquer erro referido previamente.

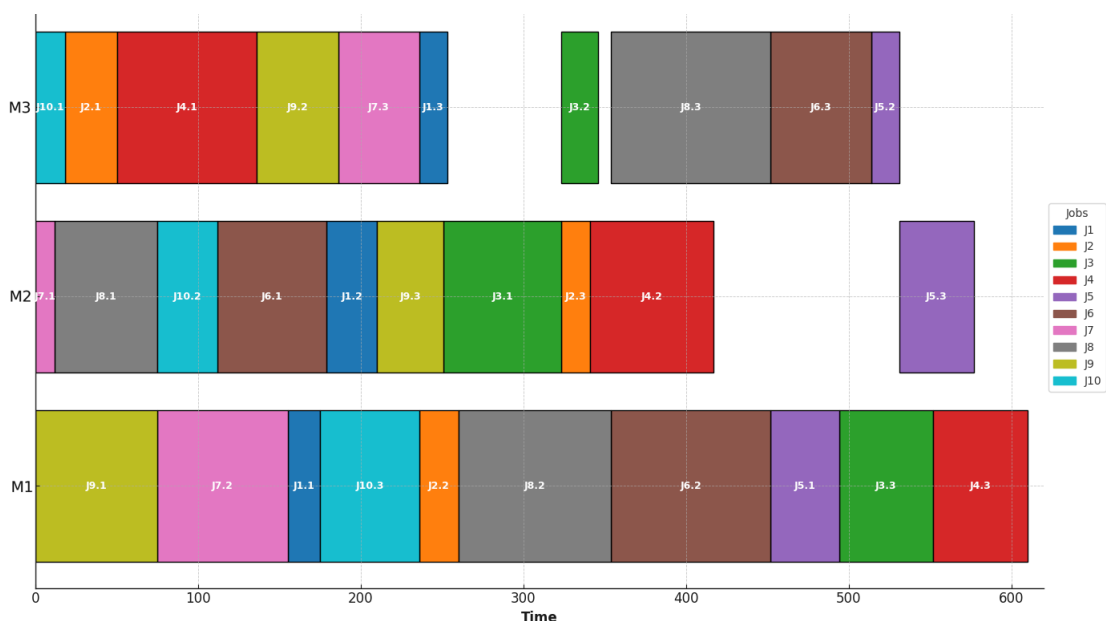


Figura 29 - Gráfico Gantt gerado pelo GPT o3 para [10Jx3M]

Tabela 17 - Programa gerado pelo GPT o3 para [10x3M]

Tarefa	Operação	Máquina	Início	Fim	Duração
J7	1	M2	0	12	12
J9	1	M1	0	75	75
J10	1	M3	0	18	18
J8	1	M2	12	75	63
J2	1	M3	18	50	32
J4	1	M3	50	136	86
J7	2	M1	75	155	80
J10	2	M2	75	112	37
J6	1	M2	112	179	67
J9	2	M3	136	186	50
J1	1	M1	155	175	20
J10	3	M1	175	236	61
J1	2	M2	179	210	31
J7	3	M3	186	236	50
J9	3	M2	210	251	41
J1	3	M3	236	253	17
J2	2	M1	236	260	24
J3	1	M2	251	323	72
J8	2	M1	260	354	94
J2	3	M2	323	341	18
J3	2	M3	323	346	23
J4	2	M2	341	417	76
J6	2	M1	354	452	98
J8	3	M3	354	452	98
J5	1	M1	452	494	42
J6	3	M3	452	514	62
J3	3	M1	494	552	58
J5	2	M3	514	531	17
J5	3	M2	531	577	46
J4	3	M1	552	610	58

Comparando a Tabela 17 e o gráfico Gantt desta solução com a Tabela 16 e o gráfico Gantt da solução ótima gerada pelo GPT o4-mini é possível identificar múltiplas diferenças na ordem em que as operações foram introduzidas, no entanto, em ambas as soluções o processamento das operações termina a 610 u.t. com a máquina M_1 totalmente ocupada 100% do tempo.

Ao solicitar a metodologia de resolução deste problema e de analisar a “fase de pensamento” do terceiro *prompt*, concluiu-se que o GPT o3 utilizou diferentes métodos de otimização para chegar a esta solução, comparando com a versão o4-mini. Este foi o procedimento utilizado:

- **Cálculo do limite inferior LI** – tal como na metodologia de resolução deste problema por parte do GPT o4-mini, o modelo GPT o3 também calculou o limite inferior para guiar a pesquisa pela solução ótima, sendo este limite definido pela carga total da máquina M_1 ($LI = 610$).
- **Foco na máquina de gargalo e SGS** – nas próximas etapas de resolução observa-se um foco especial em procurar a melhor permutação de tarefas na máquina com maior carga total M_1 , de forma a minimizar o *makespan*, gerando para cada sequência o programa inteiro, incluindo os tempos de início, de forma determinística por *Serial Schedule Generation Scheme (SGS)*, introduzindo as operações o mais cedo possível. A pesquisa apenas da sequência em M_1 é consideravelmente mais simples ($10! = 3.63e^6$ milhões de possibilidades) do que a pesquisa da sequência em todas as máquinas simultaneamente ($10! \times 10! \times 10! = 4.78e^{19}$ possibilidades).
- **Heurística NEH** – inicialmente o modelo aplicou uma heurística construtiva gulosa NEH. Este processo consiste em calcular o tempo de processamento total para cada tarefa, isto é, a soma do tempo de processamento de todas as operações pertencentes à tarefa ($P_j = \sum_{i=1}^3 p_{i,j}$), e depois ordenar as tarefas em ordem decrescente de P_j , introduzindo as primeiras da lista em M_1 na posição que menor aumente o *makespan* e gerando o programa inteiro para essa sequência seguindo a regra de SGS. Através desta heurística obteve-se um *makespan* de 670 u.t., servindo de base para o método *Branch and Bound* e de limite superior.
- **Branch and Bound** – para aprofundar a busca pela solução ótima para este problema foi utilizado o método de *Branch and Bound*, partindo de um limite superior de 670 u.t. definido pelo programa gerado pela heurística NEH. Este método procurou sistematicamente permutações de M_1 , fixando k operações na máquina, explorando diferentes sequências com as outras operações (diferentes “ramos”) e preenchendo o programa completo seguindo a regra de SGS. Se o novo programa apresenta *makespan* inferior a 670 u.t. é definido como o novo limite inferior, e todos os programas testados de seguida são comparados com estes limites. Cada programa com *makespan* superior ao limite inferior nesse momento é “podado” e o seu “ramo” eliminado da pesquisa. No final desta metodologia o modelo GPT o3 atingiu a solução ótima de 610 u.t.

O procedimento de teste dos LLMs através da resolução de problemas de Escalonamento em *Job Shop*, observado neste capítulo, foi executado para todas as instâncias apresentadas na Tabela 14, analisando o desempenho dos modelos GPT o4-mini, DeepSeek R1 e GPT o3. O *template* de *prompts* definido anteriormente no manual foi utilizado em todos os testes com objetivo de garantir estabilidade nos resultados. Também para efeito de consistência e robustez dos resultados foram realizadas três tentativas de resolução para cada problema de forma a reduzir a variabilidade e evitar valores desviantes na primeira tentativa.

3.5. Conclusão

Neste capítulo foi descrito o percurso metodológico adotado para preparar e avaliar o desempenho dos Modelos de Linguagem de Grande Escala através da aplicação de problemas de Escalonamento em ambiente Job Shop, testando a capacidade de compreensão e de resolução destes problemas por parte dos LLMs. Começou-se por apresentar a formulação geral de um problema de Escalonamento, destacando as principais características e restrições envolvidas neste tipo de problemas, como a sequência de operações pré-definida para cada tarefa e a alocação exclusiva de cada máquina a uma única operação de cada vez, complementando a análise com o uso de gráficos de Gantt e grafos disjuntivos, como ferramentas de visualização da solução. A secção seguinte explorou diferentes métodos de otimização, focando-se em regras de prioridade e heurísticas construtivas, e em métodos exatos que produzem resultados ótimos, como *Branch and Bound* e Simplex, os quais foram posteriormente aplicados à resolução de instâncias nos testes com os LLMs.

A fase seguinte centrou-se na definição do procedimento experimental utilizado para testar os LLMs, onde se estruturaram as instruções, exemplos e *inputs* fornecidos aos modelos, assim como os critérios para avaliação das respostas. Esta preparação foi fundamental para assegurar condições consistentes de teste e melhorar a comparabilidade dos resultados entre modelos. O subcapítulo 3.4 concentrou-se na análise prática, começando por uma avaliação do conhecimento geral dos modelos sobre Escalonamento, em particular sobre problemas de *Job Shop* e, de seguida, uma apresentação de várias instâncias de problemas a serem utilizados como objeto de teste para os LLMs.

Inicialmente, testaram-se os modelos GPT o4-mini e DeepSeek R1, começando por uma instância simples [3Jx3M] de 3 tarefas e 3 máquinas. Ambos os modelos demonstraram bom desempenho nesta fase, apresentando soluções coerentes e sem erros. No entanto, à medida que a complexidade das instâncias aumentou, como no caso da instância [10Jx3M] com 10 tarefas e 3 máquinas, surgiram mais dificuldades, tanto na interpretação das restrições como na geração de soluções viáveis. Foi então necessário recorrer ao modelo GPT o3, que se revelou mais estável nestas situações com uma capacidade superior de interpretação do problema e de aplicação de métodos de resolução mais elaborados, embora requeiram mais tempo de processamento para responder ao *prompt*. Este processo permitiu não só identificar as limitações e pontos fortes de cada modelo, como também compreender em maior profundidade o seu comportamento face a problemas de raciocínio estruturado e interpretação de restrições como os de *Job Shop*.

4. Resultados e Discussão

Neste capítulo serão apresentados os resultados obtidos nos testes referidos anteriormente, seguindo a metodologia de *prompting* estabelecida no Capítulo 3. No total foram utilizadas 19 instâncias distintas do problema de Escalonamento em *Job Shop*, com o objetivo de minimizar o *makespan*, permitindo avaliar o desempenho dos LLMs DeepSeek R1, GPT o4-mini e GPT o3.

Será também realizada uma análise estatística de forma a comparar os três modelos em vários parâmetros como a proximidade ao valor ótimo, a eficiência das soluções geradas e o número de tentativas até atingir o melhor resultado. Posteriormente será discutida a relevância dos resultados obtidos num enquadramento mais amplo sobre a aplicação dos LLMs na resolução de problemas de Escalonamento.

4.1. Apresentação dos Resultados

Cada um dos três LLMs em estudo foi sujeito a um máximo de três tentativas para resolver cada problema, ou até encontrar a solução ótima previamente determinada através de *Constraint Programming* no software *CPLEX Optimization Studio*. Todas as tentativas seguiram de forma rigorosa o manual de *prompting* estabelecido anteriormente para assegurar consistência no processo de interação com os modelos.

Considerando um total de 19 instâncias de teste e três modelos diferentes, é possível necessitar de realizar até 171 tentativas no total. Face a este número excessivo decidiu-se, para efeitos da análise estatística, admitir apenas, por instância e por modelo, a tentativa que obteve um *makespan* mais próximo do ótimo, desde que essa solução fosse viável e sem violações de restrições. Registou-se também o número de tentativas necessárias para se atingir essa solução.

Com o intuito de avaliar e de comparar de forma mais objetiva o desempenho dos LLMs foram definidas duas variáveis: “Proximidade” e “Eficiência”. A primeira quantifica o quão perto o *makespan* da solução obtida está do *makespan* ótimo (em valores percentuais, considerando que uma solução com *makespan* ótimo tem Proximidade igual a 100%). O segundo parâmetro relaciona a Proximidade com o tempo de processamento, com o intuito de valorizar modelos que determinem uma solução válida mais rapidamente. Estas variáveis são apresentadas através da Equação 14 e Equação 15.

$$Proximidade(\%) = \left(1 - \frac{|Makespan - Makespan_{\acute{o}timo}|}{\acute{m}ax(Makespan, Makespan_{\acute{o}timo})}\right) * 100 \quad (14)$$

$$Efici\^encia = \left(\frac{Proximidade}{Tempo_Processamento}\right) \quad (15)$$

De seguida s\^ao apresentados, na Tabela 18, os resultados obtidos de todos os testes feitos para as 19 inst\^ancias, pelos modelos DeepSeek R1, GPT o4-mini e GPT o3. Para cada teste foi registado o *Makespan* obtido, a Proximidade ao valor \acute{o}timo, o Tempo de Processamento e a Efici\^encia. Nesta tabela as solu\c7\~oes imposs\^iveis s\~ao representadas com um "X", permitindo uma leitura mais imediata do n\^umero de tentativas que os LLMs necessitaram para cada inst\^ancia. Um \acute{u}nico "X" significa que durante as tr\^es tentativas estipuladas n\~ao foi poss\^ivel alcan\c7ar uma solu\c7\~ao vi\^avel. De forma a facilitar a visualiza\c7\~ao dos dados na tabela, foi atribu\^ida uma cor a cada conjunto Inst\^ancia x Modelo: verde – *makespan* \acute{o}timo encontrado, amarelo – solu\c7\~ao com *makespan* aproximado, vermelho – n\~ao foi encontrada nenhuma solu\c7\~ao vi\^avel. Como foi referido anteriormente, apenas a solu\c7\~ao com *makespan* mais pr\^oximo do \acute{o}timo ser\^a considerada para tratamento estat\^istico dos dados, sendo que os valores de Proximidade, Tempo de Processamento e de Efici\^encia correspondem a essa solu\c7\~ao.

Tabela 18 - Resultados dos testes \^as 19 inst\^ancias

Inst\^ancia (Makespan \acute{O}timo)	Modelo	Makespan (u.t.)	Proximidade (%)	Tempo de Processamento (segundos)	Efici\^encia
3Jx3M (29 u.t.)	DeepSeek R1	29	100	277	0.361
	GPT o4-mini	29	100	35	2.857
	GPT o3	29	100	152	0.658
5Jx3M (32 u.t.)	DeepSeek R1	32	100	655	0.153
	GPT o4-mini	32	100	77	1.299
	GPT o3	32	100	401	0.249
8Jx3M (469 u.t.)	DeepSeek R1	X X 469	100	331	0.302
	GPT o4-mini	X 484 X	96.90	45	2.153
	GPT o3	469	100	169	0.592
10Jx3M (610 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	694 610	100	38	2.632
	GPT o3	610	100	282	0.355
3Jx4M (297 u.t.)	DeepSeek R1	297	100	670	0.149
	GPT o4-mini	297	100	32	3.125
	GPT o3	297	100	73	1.370
5Jx4M (355 u.t.)	DeepSeek R1	355	100	547	0.183
	GPT o4-mini	520 X	88.09	52	1.694

Instância (Makespan Ótimo)	Modelo	Makespan (u.t.)	Proximidade (%)	Tempo de Processamento (segundos)	Eficiência
		403			
	GPT o3	355	100	287	0.348
8Jx4M (508 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	508	100	550	0.182
10Jx4M (575 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	575	100	216	0.463
3Jx5M (334 u.t.)	DeepSeek R1	334	100	419	0.239
	GPT o4-mini	334	100	49	2.041
	GPT o3	334	100	124	0.806
5Jx5M (534 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X 614 X	86.97	71	1.308
	GPT o3	534	100	417	0.240
8Jx5M (578 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X 626 X	92.33	311	0.297
	GPT o3	578	100	865	0.116
10Jx5M (590 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	590	100	770	0.130
15Jx5M (926 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	926	100	509	0.196
20Jx5M (1221 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	1221	100	378	0.265
6Jx6M (538 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	554 538	100	561	0.178
7Jx7M (588 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	615 588	100	697	0.143
8Jx8M (656 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	X 669 X	98.06	336	0.292

Instância (Makespan Ótimo)	Modelo	Makespan (u.t.)	Proximidade (%)	Tempo de Processamento (segundos)	Eficiência
9Jx9M (710 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	760 X X	93.42	535	0.175
10Jx10M (903 u.t.)	DeepSeek R1	X	0	X	X
	GPT o4-mini	X	0	X	X
	GPT o3	X 969 X	93.19	798	0.117

4.2. Estatística Descritiva

Analisando os resultados dos testes para as 19 instâncias observou-se um bom desempenho do modelo GPT o3 em relação aos outros dois modelos na capacidade de gerar uma solução que apresente *makespan* ótimo. Neste parâmetro, o DeepSeek R1 e o GPT o4-mini apresentam resultados semelhantes, apesar deste último demonstrar maior eficiência.

Os dados foram introduzidos no *software* IBM SPSS Statistics para proceder com o tratamento e a análise dos resultados. Inicialmente foi efetuada uma análise descritiva para os parâmetros principais a serem avaliados, observando-se de imediato a percentagem de respostas válidas de cada modelo. O GPT o3 apresentou uma solução possível para todas as instâncias de teste, enquanto o GPT o4-mini apenas conseguiu gerar soluções para 47.4% das instâncias (9/19) e o DeepSeek R1 para 31.6% (6/19). Além de verificar quantos problemas os LLMs conseguiram encontrar soluções válidas, é também relevante analisar a distribuição de tentativas necessárias para alcançar a melhor solução, representado pelo gráfico da Figura 30.

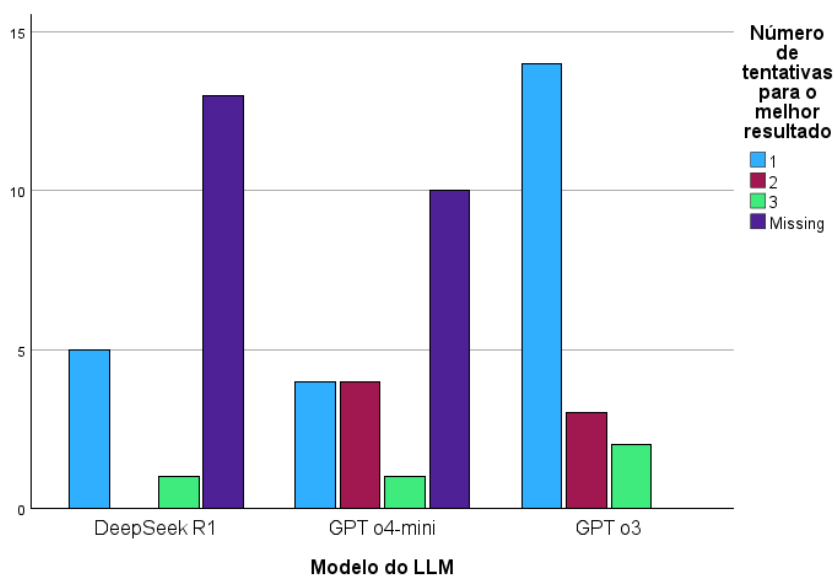


Figura 30 - Número de tentativas necessárias para cada modelo

De forma a ilustrar graficamente a discrepância entre o *makespan* gerado pelos LLMs nas soluções válidas e o *makespan* ótimo de referência para as 19 instâncias, foram elaborados os gráficos da Figura 31, Figura 32 e Figura 33. Estas representações permitem identificar de imediato o desvio de cada solução face ao ótimo e revelar padrões de comportamento dos modelos em função da complexidade das instâncias. Constituem também uma evidência visual para demonstrar a diferença entre o modelo GPT o3 e os restantes relativamente a encontrar soluções possíveis, sendo observada nestes uma limitação em resolver problemas com mais de 30 operações totais.

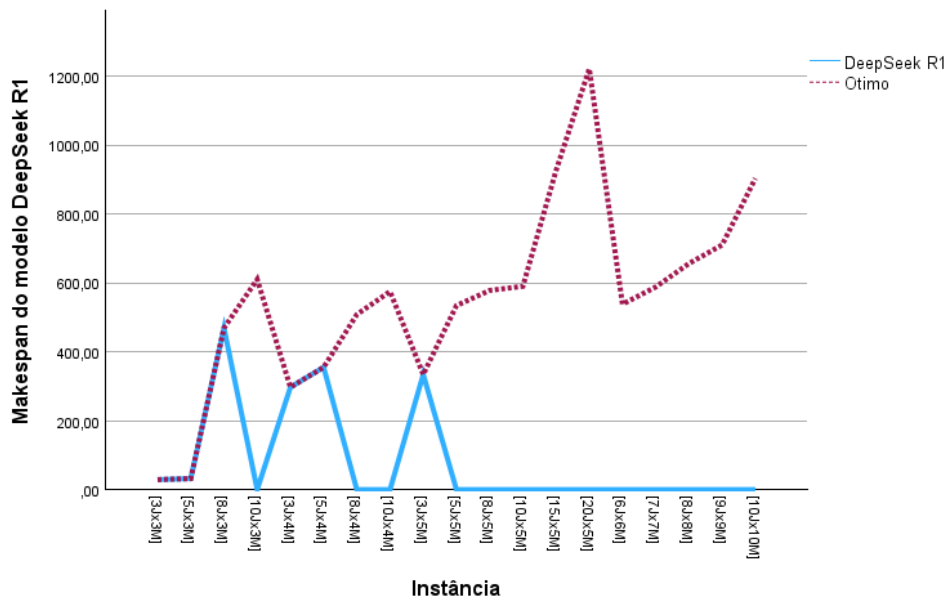


Figura 31 - Relação Makespan ótimo com Makespan gerado pelo DeepSeek R1

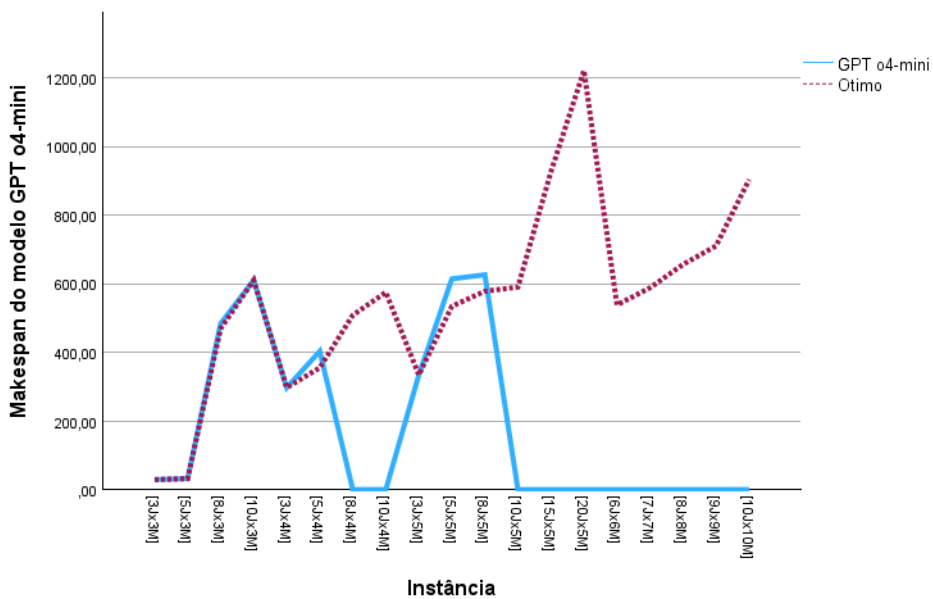


Figura 32 - Relação Makespan ótimo com Makespan gerado pelo GPT o4-mini

Resultados e Discussão

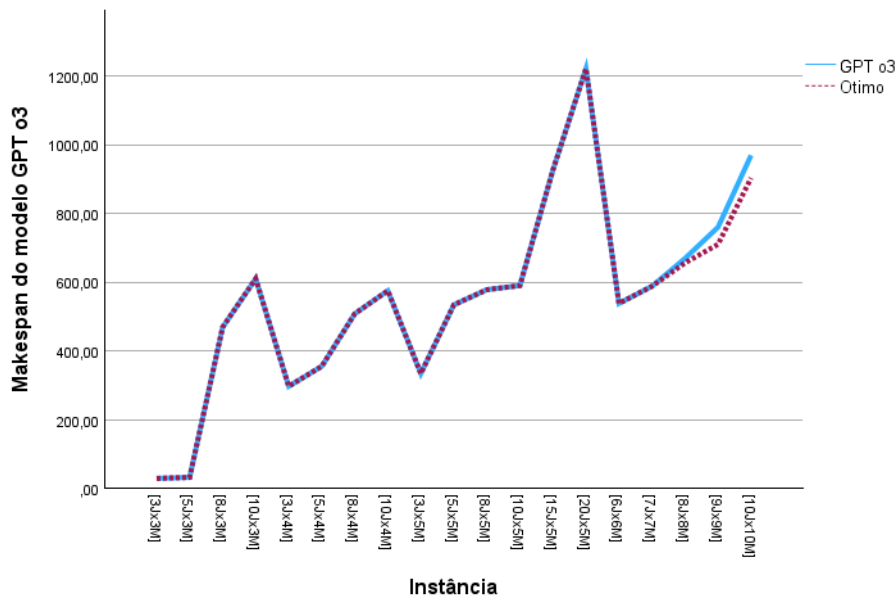


Figura 33 - Relação Makespan ótimo com Makespan gerado pelo GPT o3

Na Tabela 19,

Tabela 20, Tabela 21 e Tabela 22 estão registrados os valores médios, mínimos e máximos, e o desvio padrão, para cada modelo em relação à Proximidade, ao Tempo de Processamento, à Eficiência e ao Número de Tentativas.

De modo a penalizar os modelos que não conseguiram gerar uma solução válida para várias instâncias, a Proximidade nestes casos foi considerada 0%. Assim, o modelo DeepSeek R1 apresenta uma média de Proximidade de 31.58% apesar de todas as soluções geradas nas instâncias em que respondeu validamente (6 no total) terem sido ótimas (Proximidade=100%). Já o modelo GPT o4-mini registou uma média de 45.49% e o GPT o3 uma média de 99.19%. Observa-se, no entanto, que quando os LLMs conseguem gerar uma solução, esta tende a aproximar-se muito do *makespan* ótimo, o que pode indicar que os LLMs priorizam a obtenção de soluções ótimas, mesmo implicando um maior risco de erros.

Quanto ao Tempo de Processamento e à Eficiência, são considerados como casos omissos aqueles que não foram capazes de gerar nenhuma solução válida, sendo analisados apenas 6 casos para o DeepSeek R1, 9 casos para o GPT o4-mini e todos os 19 casos para o GPT o3. Para estas variáveis observa-se um desempenho significativamente superior por parte do modelo GPT o4-mini. Este modelo registra média de Tempo de Processamento igual a 78 segundos, com desvio padrão igual a 89s enquanto o DeepSeek R1 demora em média aproximadamente 483s, com desvio padrão de 166s, e o GPT o3 demora 427s com desvio padrão de 239s. O mesmo verifica-se para a Eficiência, com o GPT o4-mini apresentando uma média de 1.942, muito superior aos outros dois modelos (DeepSeek R1 com 0.231 e GPT o3 com 0.362). É importante salientar, no entanto, que a variação do número de testes considerados por modelo, resultante das diferentes percentagens de respostas válidas, pode influenciar os resultados. Tal situação poderá penalizar especialmente o GPT o3, uma vez que incluiu testes de maior complexidade, que naturalmente requerem mais tempo de processamento.

Tabela 19 - Análise Descritiva de Proximidade para os 3 modelos

Proximidade (%)					
Modelo	N	Média	Desvio Padrão	Mínimo	Máximo
DeepSeek R1	19	31.58	47.76	0	100
GPT o4-mini	19	45.49	49.40	0	100
GPT o3	19	99.19	2.12	93.19	100

Tabela 20 - Análise Descritiva de Tempo de Processamento para os 3 modelos

Tempo de Processamento (s)					
Modelo	N	Média	Desvio Padrão	Mínimo	Máximo
DeepSeek R1	6	483	166	277	670
GPT o4-mini	9	78	89	32	311
GPT o3	19	427	239	73	865

Tabela 21 - Análise Descritiva de Eficiência para os 3 modelos

Eficiência					
Modelo	N	Média	Desvio Padrão	Mínimo	Máximo
DeepSeek R1	6	0.231	0.086	0.149	0.361
GPT o4-mini	9	1.942	0.885	0.297	3.125
GPT o3	19	0.362	0.312	0.116	1.370

Tabela 22 - Análise Descritiva de Número de Tentativas para os 3 modelos

Número de Tentativas					
Modelo	N	Média	Desvio Padrão	Mínimo	Máximo
DeepSeek R1	6	1.33	0.82	1	3
GPT o4-mini	9	1.67	0.71	1	3
GPT o3	19	1.37	0.68	1	3

Foi ainda feita uma análise da variação da Proximidade e da Eficiência conforme o número de máquinas, para os três modelos, revelando-se particularmente útil para visualizar o efeito que o aumento da complexidade dos problemas tem nestes parâmetros. Durante uma fase de testes inicial, observou-se que para os problemas com seis ou mais máquinas, tanto o modelo

Resultados e Discussão

DeepSeek R1 como o GPT o4-mini demonstraram dificuldades consideráveis em gerar soluções válidas, mesmo quando os problemas apresentavam um número reduzido de tarefas. Por esta razão verifica-se nos gráficos da Figura 34 e da Figura 35 valores nulos para os dois modelos em problemas destas dimensões, centrando-se assim a análise na variação da Proximidade e da Eficiência do modelo GPT o3. Através destes gráficos é evidente uma diminuição na qualidade das soluções, e até mesmo da capacidade de encontrar soluções válidas com o aumento da complexidade das instâncias, e também uma diminuição da Eficiência, com a exceção da instância [8x8M] em que o modelo GPT o3 encontrou uma solução próxima ao ótimo num espaço de tempo relativamente mais curto.

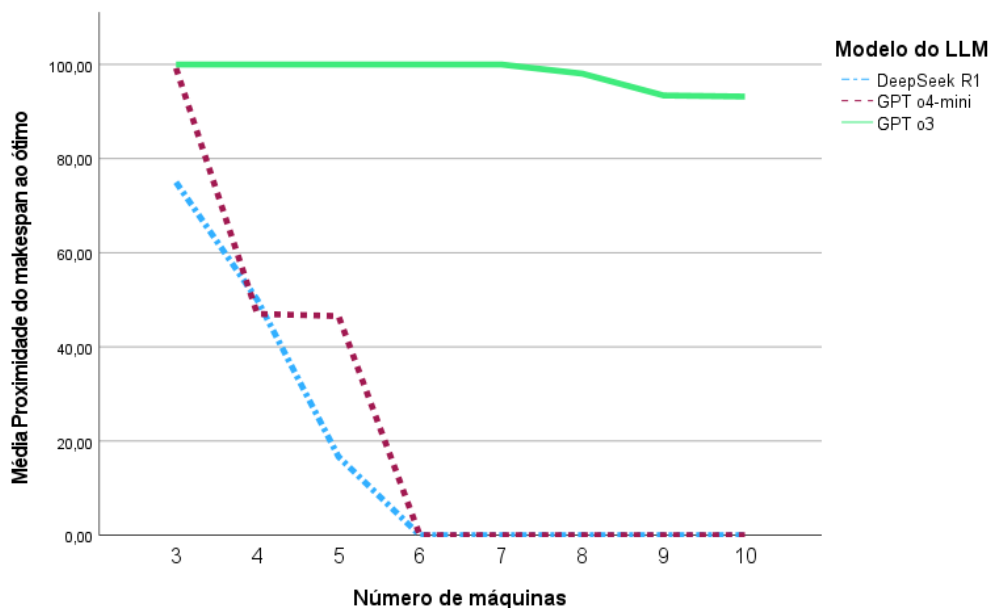


Figura 34 - Variação da Proximidade com o Número de Máquinas para os 3 modelos

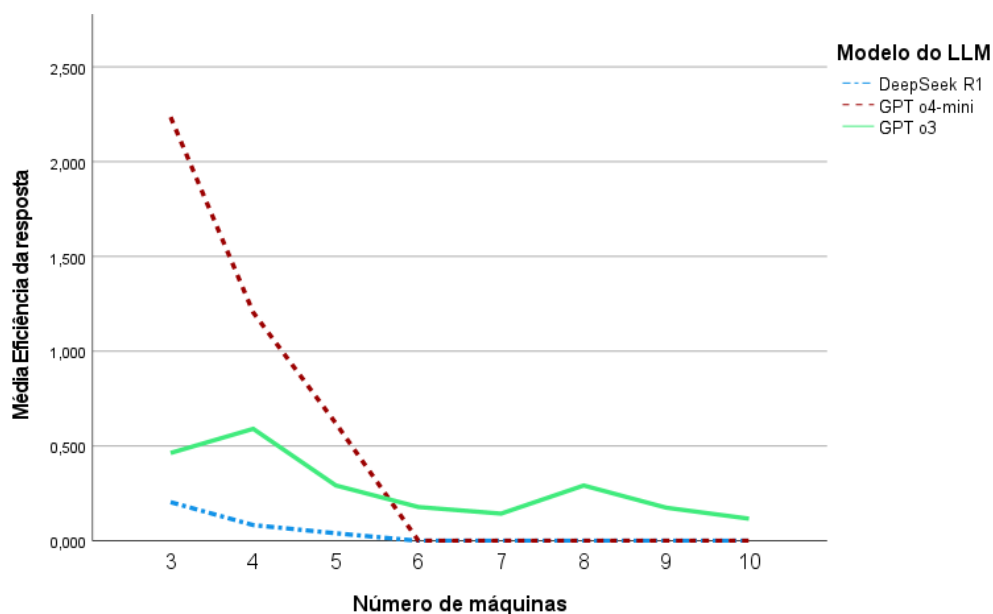


Figura 35 - Variação da Eficiência com o Número de Máquinas para os 3 modelos

4.3. Inferência Estatística

Concluída a análise descritiva dos resultados, recorreu-se a métodos de inferência estatística com o objetivo de avaliar se as diferenças observadas entre os modelos eram estatisticamente significativas. Esta abordagem permite ir para além da simples observação dos valores, apoiando-se em testes que consideram a variabilidade dos dados e possibilitam tirar conclusões mais robustas sobre o desempenho comparado dos modelos. A inferência estatística vai incidir principalmente sobre as seguintes variáveis: Proximidade, Tempo de Processamento, Eficiência e Número de Tentativas.

Para fundamentar a escolha dos testes estatísticos a efetuar, consideram-se três conjuntos de dados independentes, cada um com 19 elementos no máximo. No caso das variáveis Tempo de Processamento, Eficiência e Número de Tentativas, são consideradas apenas as instâncias com resultados válidos: 6 para o DeepSeek R1, 9 para o GPT o4-mini e 19 para o GPT o3).

Teste de Normalidade

Como estas amostras são relativamente pequenas ($n \leq 30$), foi necessário, inicialmente, verificar a normalidade para cada variável em estudo através do teste de Shapiro-Wilk. Este teste é adequado a amostras mais pequenas, sendo reconhecido como mais sensível do que alternativas como o teste de Kolmogorov-Smirnov. As hipóteses analisadas por este teste foram as seguintes:

- H_0 : A variável “x” segue distribuição normal
- H_1 : A variável “x” não segue distribuição normal

Com “x” representando a variável a ser testada no momento, incluindo Proximidade, Tempo de Processamento, Eficiência e Número de Tentativas.

A Tabela 23 mostra os resultados do teste de normalidade de Shapiro-Wilk, onde Graus de Liberdade representa o tamanho da amostra, Estatística W é um valor representativo de quanto os dados se desviam da normalidade, variando entre 0 e 1, sendo que quanto mais próximo o valor for de 1 mais normal é a distribuição, e *p-value* apresenta o valor mais importante para este teste. Para cada amostra de dados, e para um grau de confiança de 95%, se o *p-value* for maior que 0.05 não há inferência estatística que permita rejeitar H_0 , isto é, considera-se que a amostra segue uma distribuição normal. Deste modo, destaca-se na tabela as amostras com distribuição normal a verde, e as amostras que não seguem distribuição normal a vermelho.

A verificação da normalidade foi essencial para determinar o tipo de testes estatísticos mais adequados. Os resultados indicaram que a maioria dos conjuntos de dados obteve um *p-value* inferior a 5%, ou seja, não segue uma distribuição normal, embora se tenham verificado algumas exceções. Face a esta predominância de não normalidade nos conjuntos, e tendo em conta que as amostras em análise são independentes entre si (correspondendo aos diferentes modelos), optou-se pela utilização de testes não paramétricos.

Tabela 23 - Resultados do teste de Normalidade (Shapiro-Wilk)

Teste de Normalidade	Modelo	Estatística W	Graus Liberdade	<i>p-value</i>
Proximidade	DeepSeek R1	0.591	19	<0.001
	GPT o4-mini	0.673	19	<0.001
	GPT o3	0.429	19	<0.001
Tempo de Processamento	DeepSeek R1	0.911	6	0.443
	GPT o4-mini	0.537	9	<0.001
	GPT o3	0.956	19	0.496
Eficiência	DeepSeek R1	0.902	6	0.383
	GPT o4-mini	0.968	9	0.873
	GPT o3	0.746	19	<0.001
Número de Tentativas	DeepSeek R1	0.496	6	<0.001
	GPT o4-mini	0.805	9	0.024
	GPT o3	0.593	19	<0.001

Teste de Kruskal-Wallis

Assim sendo, foi selecionado o teste de Kruskal-Wallis. Este teste permite comparar três ou mais grupos independentes e determinar se existe, entre eles, diferenças estatisticamente significativas em relação às variáveis em análise. Na Tabela 24 é possível observar os resultados do teste de Kruskal-Wallis comparando os modelos DeepSeek R1, GPT o4-mini e GPT o3 para os parâmetros de Proximidade, Tempo de Processamento, Eficiência e Número de Tentativas. Também neste teste existe um valor de estatística (H), calculado com base nos *ranks* dos dados, que mede o quanto os grupos diferem entre si. Quanto maior a estatística H, maior é a evidência de que pelo menos um grupo é diferente dos outros, sendo que a maioria destes valores presentes no teste são considerados altos. Aliado a este valor, identificou-se um *p-value* inferior a 0.001 para todos os conjuntos de dados, com exceção do Número de Tentativas, confirmando que, para um grau de confiança de 95%, é possível rejeitar a hipótese de que os modelos seguem distribuições semelhantes, ou seja, existem diferenças estatisticamente significativas entre modelos para os parâmetros de Proximidade, Tempo de Processamento e Eficiência. Quanto ao Número de Tentativas, um *p-value* de 0.327 (32.7%) indica que todos os LLMs seguem uma distribuição semelhante.

Tabela 24 - Resultados do teste Kruskal-Wallis

Teste Kruskal-Wallis	Estatística H	df	<i>p-value</i>
Proximidade	19.665	2	<0.001
Tempo de Processamento	16.552	2	<0.001
Eficiência	16.166	2	<0.001
Número de Tentativas	2.235	2	0.327

Testes Mann-Whitney

Com a confirmação da existência de diferenças significativas entre modelos relativamente às variáveis Proximidade, Tempo de Processamento e Eficiência, é relevante identificar de seguida quais os modelos que apresentam estas diferenças para cada variável, através de análises adicionais Post Hoc não paramétricas. Para tal, foi utilizado o teste de Mann-Whitney para estes três parâmetros, agrupando os LLMs em duplas para proceder à sua comparação direta. Neste teste é apresentado o posto médio para cada LLM, que indica a posição relativa dos valores nos dois conjuntos de dados, e o *p-value*, que indica se existe diferença estatisticamente significativa entre os modelos. Assim, as hipóteses analisadas pelo teste Mann-Whitney são as seguintes:

- H_0 : As distribuições dos dois LLMs são iguais
- H_1 : Existem diferenças significativas entre as distribuições dos dois LLMs

Em primeiro lugar juntou-se os modelos DeepSeek R1 (A) e GPT o4-mini (B), com os resultados do teste apresentados na Tabela 25. Relativamente à Proximidade, a um nível de confiança de 95% não existe evidência estatística para rejeitar a hipótese nula, sugerindo que ambos os modelos seguem uma distribuição semelhante. Analisando os postos médios, observa-se, no entanto, que o GPT o4-mini possui valores de Proximidade ligeiramente superiores ao DeepSeek R1. Redirecionando a atenção para o Tempo de Processamento e a Eficiência, com ambos os *p-values* inferiores a 0.05 é possível afirmar estatisticamente que existem diferenças significativas entre as distribuições de ambos os modelos. Através dos valores médios é evidente que o GPT o4-mini geralmente apresenta soluções para os problemas de Escalonamento mais rápido que o DeepSeek R1, com uma Eficiência também superior. O teste de Mann-Whitney mostrou que o modelo GPT o4-mini possui um desempenho superior ao DeepSeek R1, nomeadamente quanto à rapidez e eficiência, ainda que as soluções sejam relativamente semelhantes.

Resultados e Discussão

Tabela 25 - Resultados do teste Mann-Whitney para (DeepSeek R1 - GPT o4-mini)

Teste Mann-Whitney	N _A	Posto Médio A	N _B	Posto Médio B	<i>p</i> -value (bilateral)
Proximidade	19	18.63	19	20.37	0.579
Tempo de Processamento	6	12.33	9	5.11	0.002
Eficiência	6	3.83	9	10.78	0.003

De seguida realizou-se o teste Mann-Whitney para o par de LLMs DeepSeek R1 (A) e GPT o3 (C), apresentando os valores na Tabela 26. Quando analisada a Proximidade, a um nível de confiança de 95% foram encontradas evidências estatísticas para rejeitar a hipótese nula, isto é, existem diferenças estatisticamente significativas entre os dois modelos. Como o posto médio do GPT o3 é muito superior ao do DeepSeek R1, é seguro afirmar que o primeiro obtém soluções válidas e ótimas mais regularmente do que o segundo. Com ambos os *p*-values do Tempo de Processamento (0.567) e da Eficiência (0.611) superiores a 0.05, não existe evidência estatística para rejeitar a hipótese nula. Os dois LLMs seguem distribuições semelhantes, no entanto, observando os valores dos postos médios, conclui-se que o GPT o3 apresenta soluções um pouco mais rapidamente que o DeepSeek R1. Determinou-se que a grande diferença entre estes modelos é a Proximidade das soluções ao ótimo.

Tabela 26 - Resultados do teste Mann-Whitney para (DeepSeek R1 - GPT o3)

Teste Mann-Whitney	N _A	Posto Médio A	N _C	Posto Médio C	<i>p</i> -value (bilateral)
Proximidade	19	13.47	19	25.53	<0.001
Tempo de Processamento	6	14.50	19	12.53	0.567
Eficiência	6	11.67	19	13.42	0.611

Por último realizou-se o teste Mann-Whitney para os LLMs GPT o4-mini (B) e GPT o3 (C). Na Tabela 27 observa-se que todos os *p*-values são inferiores a 0.001, por isso, a um nível de confiança de 95%, há evidências estatísticas para rejeitar a hipótese nula. Isto quer dizer que os dois LLMs seguem distribuições significativamente diferentes quando comparados os conjuntos de dados das três medidas de desempenho. Analisando os postos médios retirou-se a informação de que o modelo GPT o3 gera soluções válidas e ótimas mais regularmente que o GPT o4-mini, enquanto este gera as respostas muito mais rapidamente. Conclui-se então que entre os dois modelos da Open AI analisados neste estudo, o GPT o3 é muito mais eficaz, com potencial para resolver problemas mais complexos, ainda que seja necessária uma fase de “pensamento” mais longa, enquanto que o GPT o4-mini é um modelo eficiente para problemas de menor dimensão.

Tabela 27 - Resultados do teste Mann-Whitney para (GPT o4-mini - GPT o3)

Teste Mann-Whitney	N _B	Posto Médio B	N _C	Posto Médio C	<i>p</i> -value (bilateral)
Proximidade	19	13.00	19	26.00	<0.001
Tempo de Processamento	9	5.89	19	18.58	<0.001
Eficiência	9	23.11	19	10.42	<0.001

4.4. Discussão de Resultados

O rápido desenvolvimento e aperfeiçoamento dos Modelos de Linguagem de Grande Escala nos últimos anos têm demonstrado um elevado potencial na otimização dos processos industriais, incluindo o Escalonamento das Operações. Neste trabalho foi analisado o comportamento dos LLMs na resolução de problemas de escalonamento industriais, tendo-se definido, para esse propósito, um conjunto de 19 instâncias de Escalonamento em *Job Shop*, para as quais os LLMs procuraram gerar calendários operacionais que minimizassem o *makespan*.

O estudo incidiu sobre o GPT o4-mini, um modelo muito utilizado da OpenAI com elevada capacidade de raciocínio, disponível na versão gratuita do ChatGPT, e o DeepSeek R1, um modelo emergente também com ótima capacidade de raciocínio e acessível de forma gratuita na respetiva plataforma, do qual várias comparações com o ChatGPT foram feitas após a sua introdução no mercado. Foi também analisado o desempenho de um modelo da versão Pro do ChatGPT, o GPT o3, para comparar com os modelos de acesso livre. Com o objetivo de avaliar não só a eficácia dos LLMs, mas também a rapidez e a variabilidade das repostas, foram realizadas análises estatísticas dos parâmetros: Proximidade do *makespan* ao ótimo, Tempo de Processamento para gerar a resposta, Eficiência da resposta e Número de Tentativas necessárias para gerar o melhor *makespan*.

Rapidamente foi identificada uma semelhança na complexidade de instâncias às quais os modelos GPT o4-mini e DeepSeek R1 conseguiram gerar soluções viáveis, mais concretamente observando um aumento de dificuldade a partir das instâncias que contêm 25-35 operações e não sendo capaz de obter soluções possíveis para instâncias de complexidade superior. Ainda assim, o GPT o4-mini conseguiu apresentar soluções viáveis para 9 problemas enquanto o DeepSeek R1 apenas apresentou 6. No entanto, todas as soluções viáveis geradas por este último concluíram em *makespans* ótimos, o que pode indicar uma tendência para privilegiar a obtenção do melhor *makespan* possível, mesmo aumentando o risco de cometer erros e gerar soluções impossíveis, enquanto o GPT o4-mini mostrou múltiplas vezes soluções aproximadas. Destaca-se entre os dois modelos uma diferença significativa no Tempo de Processamento, com o GPT o4-mini demonstrando ser um modelo consideravelmente eficiente, gerando respostas em um minuto ou menos. Já o DeepSeek R1 apresentou os valores de Tempo de Processamento mais elevados, demorando aproximadamente 8 minutos em média na fase de pensamento.

Resultados e Discussão

Já o modelo GPT o3 revelou resultados muito diferentes. Todas as instâncias utilizadas foram resolvidas com sucesso por este modelo, encontrando o *makespan* ótimo para 16 das 19 instâncias e um *makespan* aproximado para os restantes 3, mesmo para instâncias com 100 operações no total. Isto evidencia a elevada eficácia deste modelo. A sua desvantagem em relação ao GPT o4-mini é claramente o Tempo de Processamento, que apresenta valores próximos dos do modelo DeepSeek R1, com uma média de aproximadamente 7 minutos. No entanto, este valor pode ser justificado pela presença de instâncias consideravelmente mais complexas na base de dados dos testes efetuados pelo GPT o3. Na verdade observa-se uma rapidez de resposta superior ao DeepSeek R1 para as instâncias mais simples, com um aumento do Tempo de Processamento para as instâncias mais complexas às quais os outros dois modelos não conseguiram apresentar solução viável, culminando numa redução da eficiência média do GPT o3.

5. Conclusão

Neste capítulo é feita uma síntese das informações extraídas de todo o processo de teste dos Modelos de Linguagem de Grande Escala através das 19 instâncias de Escalonamento, desde a fase de preparação e criação de um manual de *prompting* até à análise estatística dos resultados dos testes. Posteriormente introduz-se uma discussão sobre dificuldades durante a realização do trabalho, limitações inerentes ao tema do estudo, e possíveis caminhos a seguir para prosseguir e complementar este estudo.

5.1. Conclusões Finais

O Escalonamento das Operações é um problema central na área da gestão da produção, uma vez que tem como objetivo organizar um conjunto de tarefas a serem realizadas por recursos limitados de forma a otimizar determinados objetivos, como a minimização do tempo de conclusão de todas as tarefas (*makespan*). É um problema dinâmico, habitualmente, de elevada complexidade, devido ao número de variáveis e de restrições que pode apresentar, como o número de máquinas disponíveis e o número de tarefas que precisam de ser processadas, possíveis interrupções do processo, datas pré-definidas de conclusão das atividades e restrições de precedência.

Os ambientes em *Job Shop* aumentam ainda mais a dificuldade do problema de Escalonamento, ao introduzir rotas diferentes de processamento das tarefas, pré-determinadas, ao longo das máquinas. É este tipo de problema de Escalonamento que foi utilizado durante a realização deste trabalho para testar as capacidades dos LLMs, permitindo uma avaliação rigorosa num contexto mais complexo.

A avaliação das capacidades dos Modelos de Linguagem de Grande Escala constitui um tópico de extrema relevância, devido ao surgimento recente destas tecnologias e à rapidez a que estão a ser atualizadas e aprimoradas. Observa-se que, através do espírito competitivo, as empresas desenvolvedoras de modelos de IA estão num constante processo de incorporação de novas funcionalidades, como a possibilidade de gerar imagens ou de realizar pesquisas a fundo na *internet*, e também de melhorar o desempenho em aspetos já presentes, como a geração de código ou o raciocínio matemático. Este contínuo progresso tecnológico atrai cada vez mais utilizadores, quer pessoas no dia-a-dia que recorrem aos modelos para fins pessoais, quer empresas a procurar otimizar os seus processos.

Conclusão

Neste contexto, o interesse das empresas em aplicar LLMs à resolução de problemas industriais, e em particular ao Escalonamento das Operações, tem vindo a crescer. A capacidade destes modelos em lidar com dados diversificados, interpretar restrições complexas e propor soluções otimizadas oferece novas possibilidades para a gestão da produção. Deste modo, o propósito deste trabalho centrou-se na avaliação do desempenho e da viabilidade da utilização de LLMs na resolução de problemas de Escalonamento das Operações, analisando não só a capacidade de gerar soluções válidas e otimizadas, mas também a consistência dos resultados, a rapidez no processamento das respostas, e possíveis tendências na escolha dos métodos de otimização pelos modelos. Adicionalmente, procurou-se examinar a dificuldade de interação do utilizador com os modelos e realizada uma comparação detalhada entre os três modelos escolhidos para este estudo.

Para tal selecionou-se dois modelos amplamente utilizados à data do trabalho, o GPT o4-mini e o DeepSeek R1, e um modelo de desempenho superior, disponibilizado numa plataforma de subscrição, para explorar o potencial dos LLMs na resolução de problemas mais complexos. Seguidamente definiu-se um conjunto de 19 instâncias de Escalonamento das Operações em ambiente de *Job Shop*, adaptadas de instâncias apresentadas nos trabalhos de *Fisher e Thompson*, de *Lawrance*, e de *Storer, Wu e Vaccari* [107], e estabeleceu-se um *template* de como interagir com os modelos. Deu-se então início à fase de testes onde foram realizadas até três tentativas de resolução das instâncias por parte dos modelos, com o objetivo de alcançar o menor *makespan* possível. Foi utilizado simultaneamente *Constraint Programming*, através do software *IBM ILOG CPLEX Optimization Studio*, para determinar o *makespan* ótimo para cada instância, de forma a comparar os valores gerados pelos diferentes LLMs.

Através deste processo confirmou-se que os três LLMs possuem um elevado nível de conhecimento sobre problemas de Escalonamento das Operações. Quanto à resolução dos problemas, observou-se um evidente desempenho superior por parte do modelo GPT o3, nomeadamente pela Proximidade ao *makespan* ótimo e pela taxa de soluções válidas, tendo raramente cometido erros ou ignorado restrições, apesar de necessitar de um Tempo de Processamento considerável para gerar as respostas. Já os modelos GPT o4-mini e DeepSeek R1 demonstram ser mais limitados, não tendo conseguido apresentar soluções válidas para os problemas mais complexos, sobretudo por quebrar as restrições de precedência (colocando o processamento de uma operação posterior ($O_{(i+1),j}$) antes de uma operação precedente ($O_{i,j}$) da mesma tarefa). O ponto forte que se pode identificar no GPT o4-mini é a sua eficiência, apresentando um Tempo de Processamento muito inferior. Desta forma, se o objetivo do utilizador for a resolução rápida de problemas simples de Escalonamento, o modelo GPT o4-mini cumpre adequadamente a função, mas se quiser assegurar que obtém respostas de melhor qualidade, ou apenas estiver a tratar de problemas mais complexos, o GPT o3 será a melhor escolha. Verificou-se ainda que o número de máquinas na instância tende a ter maior impacto na complexidade do problema e, conseqüentemente, na dificuldade do LLM em resolvê-lo, do que o número de tarefas. Entre duas instâncias com o mesmo número total de operações, os LLMs parecem demonstrar pior desempenho naquela que envolve um maior número de máquinas.

Durante a fase de teste dos LLMs, observou-se um elevado nível de variabilidade nas respostas. Às vezes um modelo apresentava uma solução impossível para um problema, ignorando várias restrições impostas, e de seguida, diante do mesmo *prompt*, num chat novo, era capaz de alcançar uma solução ótima. Esta imprevisibilidade dos modelos de linguagem pode ser explicada em parte pelo processo contínuo de escolha da palavra seguinte na resposta gerada pelo modelo, que envolve uma distribuição de probabilidades. Escolhas diferentes podem levar a respostas diferentes, selecionando, por exemplo, métodos de otimização distintos. Este detalhe é muito importante para a obtenção de um *makespan* melhor, visto que vários métodos de otimização são determinísticos, levando a soluções idênticas sempre que são aplicados ao mesmo problema. Observou-se que os modelos estudados tendem a utilizar heurísticas para resolver os problemas, como o SPT (*Shortest Processing Time*) e o algoritmo de Giffler-Thompson. Este comportamento poderá estar relacionado com a menor complexidade computacional associada às heurísticas. Adicionalmente, como os LLMs não têm acesso a bibliotecas comuns de otimização (como PuLP ou OR-tools) para resolver os problemas de Escalonamento, não conseguem aplicar diretamente métodos exatos como *Branch and Bound* ou *Simplex*. Em vez disso, foi identificado por vezes o modelo a simular a lógica do método *Branch and Bound*, determinando primeiro um calendário de operações, estimando limites inferiores, explorando o espaço de soluções para um número limitado de iterações e podendo os ramos que apresentam valores fora dos limites. Ao contrário do *Branch and Bound*, esta metodologia não garante uma solução ótima, mas ajuda a aproximar a solução inicial do ótimo.

5.2. Limitações e Trabalhos Futuros

A realização deste trabalho revelou várias limitações associadas ao tema de estudo que podem afetar os resultados obtidos. Um dos fatores mais impactantes nesta análise foi o esforço necessário para assegurar resultados estáveis, devido à elevada variabilidade dos LLMs. Este fator mostrou-se difícil de controlar em várias situações: quando o modelo tentava procurar, na *internet*, uma instância tipicamente estudada que se aproximasse à do problema colocado, com o objetivo de “encontrar” o *makespan* ótimo ao invés de o “determinar”, mesmo quando era pedido para não o fazer; quando o modelo garantia que a solução apresentada otimizava completamente o problema, sem margem para melhoria, apesar do *makespan* não corresponder ao ótimo; quando o modelo indicava que não existiam erros presentes na solução gerada, embora seja possível identificar ordens das operações erradas ou sobreposição de operações. Constatou-se que, ao trabalhar com LLMs, é essencial analisar criticamente a resposta completa, uma vez que estes modelos podem apresentar incoerências ou descrever procedimentos diferentes daqueles que efetivamente aplicaram.

A qualidade da análise estatística pode também ter sido afetada pela diferença no número de instâncias com resultados válidos, isto é, as instâncias que conseguiram ser otimizadas pelos modelos, através de calendários das operações possíveis. No tratamento de dados, para os parâmetros Tempo de Processamento, Eficiência e Número de Tentativas, consideraram-se 6 testes válidos para o modelo DeepSeek R1, 9 testes válidos para o GPT o4-mini e todos os 19 testes para o GPT o3. Como o GPT o3 conseguiu encontrar soluções viáveis para as instâncias mais complexas, para as quais é necessário, à partida, maior Tempo de Processamento para o

Conclusão

LLM gerar a resposta, a média deste parâmetro aumentou, diminuindo também a Eficiência do modelo, situação que não se verifica com os outros dois modelos, que apenas conseguiram resolver instâncias simples. Esta abordagem é, no entanto, alterada para o parâmetro de Proximidade, para assegurar uma melhor representação das capacidades dos LLMs. Neste caso são considerados 19 testes válidos para todos os modelos, com Proximidade igual a 0 para as soluções impossíveis, de forma a penalizar os modelos com menor taxa de sucesso.

Futuramente será relevante analisar o desempenho de outros LLMs, incluindo modelos atuais que tenham as suas características melhoradas de forma a serem capazes de resolver problemas de Escalonamento das Operações e ainda modelos que sejam criados no futuro, e comparar com o DeepSeek R1, o GPT o4-mini e o GPT o3. Poderão também ser consideradas mais instâncias, para analisar em maior detalhe as oscilações dos parâmetros com o aumento de tarefas ou com o aumento de máquinas, incluindo instâncias de maior complexidade que permitem testar os limites dos modelos mais robustos.

Uma consequência da recente da utilização e estudo de LLMs é o facto da aplicação destes modelos na resolução de problemas de Escalonamento das Operações ser um campo ainda pouco explorado e documentado na literatura científica. Esta escassez de trabalhos publicados tornou-se evidente durante a Análise Bibliométrica e a Revisão do Estado da Arte, refletindo a natureza emergente do tema e a limitação de referências diretamente relacionadas. No entanto, este cenário representa uma oportunidade para investigações futuras aprofundarem o estudo, à medida que novas publicações venham a enriquecer a base bibliográfica.

Referências

- [1] M. Fera, F. Fruggiero, A. Lambiase, G. Martino e M. Elena, "Production Scheduling Approaches for Operations Management," em *Operations Management*, 2013.
- [2] E. K. Demeulemeester e W. S. Herroelen, *Project Scheduling: A Research Handbook*, 2002.
- [3] P. Brucker, *Scheduling Algorithms*, Springer Berlin, 2007.
- [4] I. Harjunkoski, C. T. Maravelias, P. Bongers, P. M. Castro, S. Engell, I. E. Grossmann, J. Hooker, C. Méndez, G. Sand e J. Wassick, "Scope for industrial applications of production scheduling models and solution methods," *Computers and Chemical Engineering*, vol. 62, pp. 161-193, 2014.
- [5] D. R. Sule, *Industrial Scheduling*, PWS Publishing Co., 1996.
- [6] J. Błażewicz, K. H. Ecker, E. Pesch, G. Schmidt e J. Węglarz, *Handbook on Scheduling*, Springer Berlin, Heidelberg, 2007.
- [7] A. B. G. S. e. Santos, "Auto-parametrização de meta-heurísticas para problemas de escalonamento em ambiente industrial," PhD Dissertation, Universidade do Minho, Guimarães, Portugal, 2020.
- [8] H. S. Kasana e K. D. Kumar, *Introductory Operations Research: Theory and Applications*, Springer Berlin, Heidelberg, 2004.
- [9] K. R. Baker e D. Trietsch, *Principles of Sequencing and Scheduling*, Wiley, 2009.
- [10] R. W. Conway, W. L. Maxwell e L. W. Miller, *Theory Of Scheduling*, 1967.
- [11] S. French, *Sequencing and scheduling: An introduction to the mathematics of the job-shop*, Wiley, 1982.
- [12] R. L. Graham, E. L. Lawler, J. K. Lenstra e A. H. G. Rinnooy Kan, *Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey*, Elsevier, 1979.
- [13] M. L. Pinedo, *Scheduling, Theory, Algorithms and Systems*, Springer New York, NY, 2008.
- [14] Y.-H. Chang, C.-H. Liu e Y. S. D., "Scheduling for the Flexible Job-Shop Problem with a Dynamic Number of Machines Using Deep Reinforcement Learning," *Information*, vol. 15, nº 2, 2024.
- [15] T. Alves, I. Tommelein e G. Ballard, "Value Stream Mapping for Make-to-Order Products in a Job Shop Environment," em *Construction Research Congress 2005: Broadening Perspectives - Proceedings of the Congress*, San Diego, CA, 2005.
- [16] X. Wen, H. Zhang, H. Li, H. Wang, W. Ming, Y. Zhang e L. Zhang, "Fusion Q-Learning Algorithm for Open Shop Scheduling Problem with AGVs," *Mathematics*, vol. 12, nº 3, 2024.

Referências

- [17] C. F. C. F. Shukla, "The state of the art in intelligent real-time FMS control: a comprehensive survey," *Journal of Intelligent Manufacturing*, nº 7, pp. 441-455, 1996.
- [18] V. Suresh e D. Chaudhuri, "Dynamic scheduling-A survey of research," *International Journal of Production Economics*, pp. 53-63, 1993.
- [19] D. Ouelhadj e S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, nº 12, pp. 417-431, 2009.
- [20] D. Ivanov e B. Sokolov, "Dynamic supply chain scheduling," *Journal of Scheduling*, nº 15, pp. 201-216, 2012.
- [21] N. Mulani e H. Lee, "New Business Models for Supply Chain Excellence," em *Achieving Supply Chain Excellence Through Technology*, Montgomery Research, Inc., 2002.
- [22] E. Rich, *Artificial Intelligence*, McGraw-Hill, 1983.
- [23] D. A. Hashimoto, G. Rosman, D. Rus e O. R. Meireles, "Artificial Intelligence in Surgery: Promises and Perils," *Annals of Surgery*, vol. 268, nº 1, pp. 70-76, 2018.
- [24] J. Bughin, E. Hazan, S. Ramaswamy, M. Chui, T. Allas, P. Dahlström e N. T. M. Henke, "Artificial Intelligence The Next Digital Frontier," 2017.
- [25] E. M. A. Brynjolfsson, "What's Driving the Machine Learning Explosion?," 18 Julho 2017. [Online]. Available: <https://hbr.org/2017/07/whats-driving-the-machine-learning-explosion>. [Acedido em 15 Dezembro 2024].
- [26] E. Topol, *Deep Medicine How Artificial Intelligence Can Make Healthcare Human Again*, Basic Books, 2019.
- [27] T. G. Myers, P. N. Ramkumar, B. F. Ricciardi, K. L. Urish, J. Kipper e C. Ketonis, "Artificial Intelligence and Orthopaedics: An Introduction for Clinicians," *Journal of Bone and Joint Surgery*, vol. 102, nº 9, pp. 830-840, 2020.
- [28] E. Brynjolfsson e A. McAfee, "The Business of Artificial Intelligence," 18 Julho 2017. [Online]. Available: <https://hbr.org/2017/07/the-business-of-artificial-intelligence>. [Acedido em 7 Dezembro 2024].
- [29] A. Agrawal, J. S. Gans e A. Goldfarb, "Exploring the impact of artificial Intelligence: Prediction versus judgment," *Information Economics and Policy*, vol. 47, pp. 1-6, 2019.
- [30] M. Sharabov e G. Tsochev, "The Use of Artificial Intelligence in Industry 4.0," *Problems of Engineering Cybernetics and Robotics*, vol. 73, pp. 17-29, 2020.
- [31] W. E. Forum, "Future of Jobs Report," Suíça, 2023.
- [32] P. P. Shinde e S. Shah, "A Review of Machine Learning and Deep Learning Applications," em *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Pune, India, 2018.
- [33] I. El Naqa e M. Murphy, "What is Machine Learning?," em *Machine Learning in Radiation Oncology*, Springer, 2015, pp. 3-11.
- [34] S. Angra e S. Ahuja, "Machine learning and its applications: A review," em *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, Chirala, Andhra Pradesh, India, 2017.
- [35] M. Mohri, A. Rostamizadeh e A. Talwalkar, *Foundations of Machine Learning* second edition, MIT Press, 2018.
- [36] A. Jerry e R. Zheng, "Foundations and Trends in Machine Learning," 2024.
- [37] K. El Boucheffy e R. S. de Souza, "Chapter12 - Learning in Big Data: Introduction to Machine Learning," em *Knowledge Discovery in Big Data from Astronomy and Earth Observation*, Elsevier, 2020, pp. 225-249.

- [38] W. S. McCulloch e W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115-133, 1943.
- [39] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 49, pp. 433-460, 1950.
- [40] D. O. Hebb, *The Organization of Behaviour*, New York: John Wiley & Sons, 1949.
- [41] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, nº 6, pp. 386-408, 1958.
- [42] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, nº 3, pp. 210-229, 1959.
- [43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville e Y. Bengio, "Generative Adversarial Nets," em *Proceedings of the International Conference on Neural Information Processing Systems*, Montreal, Canada, 2014.
- [44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser e I. Polosukhin, "Attention is all you need," em *Advances in Neural Information Processing Systems 30 - 31st Annual Conference on Neural Information Processing Systems*, Long Beach, California, USA, 2017.
- [45] C. Janiesch, P. Zschech e K. Heinrich, "Machine Learning and Deep Learning," *Electronic Markets*, vol. 31, pp. 685-695, 2021.
- [46] S.-I. Ao, B. Rieger e M. A. Amouzegar, *Machine Learning and System Engineering*, Dordrecht/Heidelberg/London/New York: Springer, 2010.
- [47] B. Apolloni, A. Ghosh, F. Alpaslan e S. Patnaik, *Machine Learning and Robot Perception*, Berlin: Springer, 2005.
- [48] C. M. Bishop, *Pattern Recognition and Machine Learning*, New York: Springer, 2006.
- [49] T. J. Cleophas e A. H. Zwinderman, *Machine Learning in Medicine Part Two*, New York: Springer, 2013.
- [50] J. D. Malley, K. G. Malley e S. Pajevic, *Statistical learning for biomedical data*, Cambridge: Cambridge University Press, 2011.
- [51] S. Mitra, S. Datta, T. Perkins e G. Michailidis, *Introduction to Machine Learning and Bioinformatics*, New York: CRC Press, 2008.
- [52] A. H. Fielding, *Machine learning methods for ecological applications*, Boston: Kluwer Academic Publishers, 1999.
- [53] L. Györfi, G. Ottucsak e H. Walk, *Machine Learning For Fincancial Engineering*, London: Imperial College Press, 2012.
- [54] J. Yu e D. Tao, *Modern Machine Learning Techniques and Their Applications in Cartoon Animation Research*, Hoboken: Wiley-IEEE Press, 2013.
- [55] G. E. Hinton, S. Osindero e Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, nº 7, pp. 1527-1554, 2006.
- [56] Y. Lecun, L. Bottou, Y. Bengio e P. Haffner, "Gradient-based learning applied to document recognition," *Proceeding of the IEEE*, vol. 86, nº 11, pp. 2278-2324, 1998.
- [57] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, nº 2, pp. 179-211, 1990.
- [58] M. Arora, S. Dhawan e K. Singh, "Deep Learning Overview, Architecture, Framework & Applications," *International Journal of Latest Trends in Engineering and Technology*, vol. 10, nº 1, pp. 379-384, 2018.

Referências

- [59] M. Dixon, D. Klabjan e B. J. Hoon, "Classification-based financial markets prediction using deep neural networks," *Algorithmic Finance*, vol. 6, nº 3-4, pp. 67-77, 2017.
- [60] U. K. Kar e R. Dash, "The Future of Health and Healthcare in a World of Artificial Intelligence," *Archives in Biomedical Engineering & Biotechnology*, vol. 1, 2018.
- [61] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes e A. Mian, "A Comprehensive Overview of Large Language Models," 2024.
- [62] S. A. d. A. Arruda, "O uso de modelos de linguagem de grande escala (LLMs) no auxílio ao desenvolvimento e qualidade de software," 2024.
- [63] V. Mehta, "ChatGPT: An AI NLP Model - POV," LTIMindTree, [Online]. Available: <https://www.ltimindtree.com/wp-content/uploads/2023/02/ChatGPT-An-AI-NLP-Model-POV.pdf>. [Acedido em 30 dezembro 2024].
- [64] J. Chatterjee e N. Dethlefs, "This new conversational AI model can be your friend, philosopher, and guide ... and even your worst enemy," *Patterns*, vol. 4, nº 1, pp. 6-8, 2023.
- [65] H. H. Thorp, "ChatGPT is fun, but not an author," *Science*, vol. 379, nº 6630, p. 313, 2023.
- [66] V. Mahajan, "100+ Incredible ChatGPT Statistics & Facts in 2025," notta, 10 10 2024. [Online]. Available: https://www.notta.ai/en/blog/chatgpt-statistics?utm_source=chatgpt.com#chat-gpt-history. [Acedido em 30 12 2024].
- [67] A. Humlum e E. Vestergaard, "The Adoption of ChatGPT," Becker Friedman Institute for Economics, Chicago, USA, 2024.
- [68] J. Deng e Y. Lin, "The Benefits and Challenges of ChatGPT: An Overview," *Frontiers in Computing and Intelligent Systems*, vol. 2, nº 2, pp. 81-83, 2022.
- [69] OpenAI, "GPT-4 Technical Report," 2023.
- [70] A. Rasheed, S. A. Chasib, H. N. Alabbas, M. Zarkoosh e S. F. Abbas, "IELTS Writing Tasks Evaluation Using LLMs: A Comparative Study of ChatGPT, Claude, Copilot, and Google Gemini," em *2024 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*, El Salvador, 2024.
- [71] J. Stratton, *Copilot for Microsoft 365: Harness the Power of Generative AI in the Microsoft Apps You Use Every Day*, Charlestown, Rhode Island, USA: Apress, 2024.
- [72] M. Imran e N. Almusharraf, "Google Gemini as a next generation AI educational tool: a review of emerging educational technology," *Smart Learning Environments*, vol. 11, nº 1, pp. 1-8, 2024.
- [73] A. Robertson, "Google apologizes for 'missing the mark' after Gemini generated racially diverse Nazis," *The Verge*, 21 Fevereiro 2024. [Online]. Available: <https://www.theverge.com/2024/2/21/24079371/google-ai-gemini-generative-inaccurate-historical>. [Acedido em 27 Março 2025].
- [74] H. Hochmair, L. Juhász e T. Kemp, "Correctness Comparison of ChatGPT-4, Gemini, Claude-3, and Copilot for Spatial Tasks," *Transactions in GIS*, vol. 28, nº 7, pp. 2219-2231, 2024.
- [75] M. E. Kayaalp, R. Prill, E. A. Sezgin, T. Cong, A. Królikowska e M. T. Hirschmann, "DeepSeek versus ChatGPT: Multimodal artificial intelligence revolutionizing scientific discovery. From language editing to autonomous content generation-Redefining innovation in research and practice," *Knee Surg Sports Traumatol Arthroscopy*, 2025.

- [76] Y. Peng, B. A. Malin, J. F. Rousseau, Y. Wang, Z. Xu, X. Xu, C. Weng e J. Bian, "From GPT to DeepSeek: Significant gaps remain in realizing AI in healthcare," *Journal of Biomedical Informatics*, vol. 163, pp. 1-3, 2025.
- [77] DeepSeek-AI, "DeepSeek-V3 Technical Report," 2024.
- [78] J. Vincent, "The DeepSeek panic reveals an AI world ready to blow," *The Guardian*, 28 Janeiro 2025. [Online]. Available: <https://www.theguardian.com/commentisfree/2025/jan/28/deepseek-r1-ai-world-chinese-chatbot-tech-world-western>. [Acedido em 31 Março 2025].
- [79] J. H. Lee e J. Shin, "How to Optimize Prompting for Large Language Models in Clinical Reserach," *Korean Journal of Radiology*, vol. 25, nº 10, pp. 869-873, 2024.
- [80] Y. Sasaki, H. Washizaki, J. Li, D. Sander, N. Yoshioka e Y. Fukazawa, "Systematic Literature Review of Prompt Engineering Patterns in Software Engineering," em *IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*, Osaka, Japan, 2024.
- [81] A. Sridhar, C.-F. Lo, F. F. Xu, H. Zhu e S. Zhou, "Hierarchical Prompting Assists Large Language Model on Web Navigation," *Findings of the Association for Computational Linguistics: EMNLP 2023*, p. 10217–10244, 2023.
- [82] G. Jiang, M. Zhihao, L. Zhang e J. Chen, "Prompt engineering to inform large language model in automated building energy modeling," *Energy*, vol. 316, 2025.
- [83] L. J. Laki e Z. G. Yang, "Improving Machine Translation Capabilities by Fine-Tuning Large Language Models and Prompt Engineering with Domain-Specific Data," em *IEEE 3rd Conference on Information Technology and Data Science (CITDS)*, Debrecen, Hungary, 2024.
- [84] Z. Hu, C. Wang, Y. Shu, H.-Y. Paik e L. Zhu, "Prompt Perturbation in Retrieval-Augmented Generation based Large Language Models," em *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, Barcelona, Spain, 2024.
- [85] A. Vogelsang, "From Specifications to Prompts: On the Future of Generative LLMs in Requirements Engineering," *IEEE Software*, vol. 41, nº 5, pp. 9-13, 2024.
- [86] W. Ma, C. Yang e C. Kästner, "(Why) Is My Prompt Getting Worse? Rethinking Regression Testing for Evolving LLM APIs," em *IEEE/ACM 3rd International Conference on AI Engineering – Software Engineering for AI (CAIN)*, Lisbon, Portugal, 2024.
- [87] "Web of Science," [Online]. Available: <https://www.webofscience.com/wos/woscc/basic-search>. [Acedido em 28 dezembro 2024].
- [88] D. Jobson e Y. Li, "Investigating the Potential of Using Large Language Models for Scheduling," em *1st International Conference on AI-Powered Software (AIware)*, Porto de Galinhas, Brazil, 2024.
- [89] L. Gao, S. Ma, J. He, L. Yin, X. Kong e J. Luo, "Research on Multi-objective Optimal Scheduling Model of Industrial Text Information Based on Machine Learning," em *6th International Conference on Natural Language Processing (ICNLP)*, Xian, People's Republic of China, 2024.
- [90] A. Azab, O. Hany e F. Baki, "CAPP-GPT: A Computer-Aided Process Planning-Generative Pretrained Transformer Framework for Smart Manufacturing," *Manufacturing Letters*, vol. 41, pp. 51-62, 2024.

Referências

- [91] Z. Zheng, X. Ren, F. Xue, Y. Luo, X. Jiang e Y. You, "Response Length Perception and Sequence Scheduling: An LLM-Empowered LLM Inference Pipeline," em *37th Conference on Neural Information Processing Systems (NeurIPS)*, New Orleans, LA, 2023.
- [92] S. A. Prieto, E. T. Mengiste e B. García de Soto, "Investigating the Use of ChatGPT for the Scheduling of Construction Projects," *Buildings*, vol. 13, nº 4, 2023.
- [93] H. Y. Kwan, "User-Focused Telehealth Powered by LLMs: Bridging the Gap between Technology and Human-Centric Care Delivery," em *4th International Conference on Computer Communication and Artificial Intelligence (CCAI)*, Xian, People's Republic of China, 2024.
- [94] D. Wu, X. Wang, Y. Qiao, Z. Wang, J. Jiang, S. Cui e F. Wang, "NetLLM: Adapting Large Language Models for Networking," em *2024 ACM SIGCOMM Conference*, Sydney, Australia, 2024.
- [95] K. Papaioannou e T. D. Doudali, "The Importance of Workload Choice in Evaluating LLM Inference Systems," em *4th Workshop on Machine Learning and Systems (EuroMLSys)*, Athens, Greece, 2024.
- [96] W. Zhou, J. Wang, L. Zhu, Y. Wang e Y. Ji, "Flight Arrival Scheduling via Large Language Model," *Aerospace*, vol. 11, nº 10, 2024.
- [97] P. Jiang, H. Wang, Z. Cai, L. Gao, W. Zhang, R. Ma e X. Zhou, "SLoB: Suboptimal Load Balancing Scheduling in Local Heterogeneous GPU Clusters for Large Language Model Inference," *IEEE Transactions on Computational Social Systems*, vol. 11, nº 6, pp. 7941-7951, 2024.
- [98] S. Fernandes e H. R. Lourenço, "A Simple Optimised Search Heuristic for the Job Shop Scheduling Problem," em *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, Faro, Portugal, Springer Berlin, Heidelberg, 2008, pp. 203-218.
- [99] G. R. Heerkens, *Project Management*, McGraw Hill, 2002.
- [100] X. Wang, Z. Songping e C. Qingxin, "Unrelated parallel machine scheduling with random rework and limited preemption," *Computers & Operations Research*, vol. 177, nº 6, pp. 1-13, 2025.
- [101] W. T. Lunardi, E. G. Birgin, P. Laborie, D. P. Ronconi e H. Voos, "Mixed Integer linear programming and constraint programming models for the online printing shop scheduling problem," *Computers and Operations Research*, vol. 123, pp. 1-20, 2020.
- [102] E.-G. Talbi, *Metaheuristics - From Design To Implementation*, New Jersey, United States of America: John Wiley & Sons, 2009.
- [103] M. L. R. Varela, "Uma Contribuição para o Escalonamento da Produção baseado em Métodos Globalmente Distribuídos," Tese de Doutoramento, Minho, Portugal, 2007.
- [104] M. Mehdi, "Parallel Hybrid Optimization Methods for Permutation Based Problems," Lille, França, 2011.
- [105] H. R. Lourenço, "Job-shop scheduling: Computational study of local search and large-step optimization methods," *European Journal of Operational Research*, vol. 83, nº 2, pp. 347-364, 1995.
- [106] M. S. Nagano e H. H. Miyata, "Review and classification of constructive heuristics mechanisms for no-wait flow shop problem," *The International Journal of Advanced Manufacturing Technology*, vol. 86, pp. 2161-2174, 2016.

- [107] J. E. Beasley, "OR-Library Job shop scheduling," [Online]. Available: <https://people.brunel.ac.uk/~mastjjb/jeb/orlib/jobshopinfo.html>. [Acedido em 3 Maio 2025].

Declaração de Integridade

Declaro ter conduzido este trabalho académico com integridade. Não plagiei ou apliquei qualquer forma de uso indevido de informações ou falsificação de resultados ao longo do processo que levou à sua elaboração.

Declaro que o trabalho apresentado neste documento é original e de minha autoria, não tendo sido utilizado anteriormente para nenhum outro fim.

Declaro ainda que tenho pleno conhecimento do Código de Conduta Ética do P.PORTO.

NOME: João Pedro Fernandes Marques

ISEP, Porto, 5 de setembro de 2025

Apêndice A

Manual de *prompting* para problemas de Escalonamento das Operações

EVITAR

I want to solve an operation scheduling problem, specifically a Job Shop scheduling problem, with the objective of minimizing makespan.

This problem presents 3 jobs and 3 machines, where any job can be processed in any machine. Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine. There are some restrictions. First, each machine can only process one operation at a time. Every machine is available at the start. There are no precedence constraints between operations from different machines.

Here are the processing times (in minutes): Job 1: $t_{1,1} = 7$ (in machine 1); $t_{2,1} = 6$ (in machine 2); $t_{3,1} = 6$ (in machine 3). Job 2: $t_{1,2} = 6$ (in machine 2); $t_{2,2} = 7$ (in machine 3); $t_{3,2} = 7$ (in machine 1). Job 3: $t_{1,3} = 8$ (in machine 3); $t_{2,3} = 5$ (in machine 2); $t_{3,3} = 9$ (in machine 1)



- Um único *prompt*;
- Informação muito condensada;
- Pouca organização;
- Fraca distinção entre diferentes dados

FAZER

1º *prompt*

I have a Job Shop Scheduling Problem for you to solve. I will present the details of this problem during the next prompts and after that I will ask you to give me a solution.

2º *prompt*

*First, here are the **processing times** of the operations of every job, as well as the **machines** required to process them, in order of the pre-determined **sequence**.*

J1: (M0, 7) -> (M1, 6) -> (M2, 6).

J2: (M1, 6) -> (M2, 7) -> (M0, 7).

J3: (M2, 8) -> (M1, 5) -> (M0, 9).

Remember these times and sequences to apply to this problem

3º prompt

The objective of this Scheduling Problem is minimizing makespan. This problem presents 3 jobs and 3 machines, where any job can be processed in any machine.

There are some restrictions that must be respected at all times:

1. **Operation Sequence:** Each job has a pre-determined sequence of 3 operations and each operation should be processed in a different machine.
2. **Machine Overlap:** Each machine can only process one operation at a time.
3. **Machine Availability:** Every machine is available at the start.
4. **Precedence Constraints:** An operation can only start processing after the previous operation from the same job finishes processing. There are no precedence constraints between operations from different machines.

Using the method that is most appropriate for this case, give me an optimal solution to reduce total makespan (without giving me code to run on my PC). Focus on giving accurate information and complying with all restrictions, even if it takes more time to process the answer.



- Prompt-chaining: dividir a informação por vários prompts [];
- Melhor organização (ex. listas enumeradas) [];
- Definição clara da informação apresentada [];

Instruções extra:

➤ **Utilização constante do mesmo formato de prompt**

→ ajuda o LLM a interpretar o problema mais rapidamente (caso seja utilizado o mesmo chat)

➤ **Especificar que o LLM não deve apresentar o código para o utilizador resolver o problema**

→ evita a tendência dos modelos de orientar o utilizador sobre como resolver o problema

➤ **Pedir para priorizar soluções viáveis em vez de respostas rápidas**

→ evita que o LLM apresente uma solução com erros devido a um reduzido tempo de processamento da resposta

➤ **Confirmar sempre as respostas do modelo**

→ depois dos primeiros 2 prompts, para assegurar que o LLM compreendeu o problema corretamente, e depois do 3º prompt para identificar possíveis erros na solução, tais como:

- Contradições dentro do mesmo prompt – informações ou valores diferentes;
- Ordem incorreta – operações com tempos de início inferiores a operações precedentes;
- Sobreposições – operações da mesma tarefa a serem processadas ao mesmo tempo