NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

ANTÓNIO JOSÉ RAPOSO DO ROSÁRIO MESTRE

Licenciado em Engenharia Informática

ALGORITMOS GENÉTICOS PARA A IMPLEMENTAÇÃO DE UM PLANEAMENTO DE PRODUÇÃO INDUSTRIAL INTELIGENTE (*SMART SCHEDULING*)

DISSERTAÇÃO DE TESE DE MESTRADO DESENVOLVIDA COM A *MUVU TECHNOLOGIES*

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Setembro, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

ALGORITMOS GENÉTICOS PARA A IMPLEMENTAÇÃO DE UM PLANEAMENTO DE PRODUÇÃO INDUSTRIAL INTELIGENTE (*SMART SCHEDULING*)

DISSERTAÇÃO DE TESE DE MESTRADO DESENVOLVIDA COM A *MUVU TECHNOLOGIES*

ANTÓNIO JOSÉ RAPOSO DO ROSÁRIO MESTRE

Licenciado em Engenharia Informática

Orientador: Bruno Miguel Lopes e Silva
Diretor de R&D - Muvu Technologies | Professor Assistente Convidado - Instituto Politécnico de Leiria

Coorientador: Jorge Carlos Ferreira Rodrigues da Cruz
Professor Assistente, Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Júri

Presidente: Cláudia Alexandre Magalhães Soares
Professora Auxiliar, Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Arguente: Ricardo João Rodrigues Gonçalves
Professor Auxiliar, Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa

Orientador: Bruno Miguel Lopes e Silva
Diretor de R&D - Muvu Technologies | Professor Assistente Convidado - Instituto Politécnico de Leiria

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Setembro, 2023

Algoritmos Genéticos para a Implementação de um Planeamento de Produção Industrial Inteligente (*Smart Scheduling*)
Dissertação de Tese de Mestrado desenvolvida com a *Muvu Technologies*

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v6.10.10) [34].

Para a minha mãe, família e amigos e em memória do meu pai e dos meus avós.

Agradecimentos

Durante este período de tempo em que trabalhei nesta dissertação de tese, tive o prazer e o privilégio de conviver com um conjunto de pessoas a quem devo agradecer os seus contributos diretos e indiretos no trabalho desenvolvido.

Em primeiro lugar, gostava de agradecer aos meus orientadores Bruno Silva e Prof. Jorge Cruz, por todas as horas despendidas e ajuda prestada neste percurso. Sinto que os conselhos, sugestões, debates de ideias e dúvidas esclarecidas foram fulcrais na realização deste trabalho e que também tiveram um impacto significativo no meu desenvolvimento a nível pessoal e profissional.

Não posso deixar de agradecer a quem inicialmente me desafiou para embarcar nesta jornada. Por isso, quero agradecer ao Tiago Peralta Santos, CEO da *Muvu Technologies*, pela sua disponibilidade em ajudar e pelas suas virtudes profissionais e humanas que influenciaram o meu crescimento a nível profissional e pessoal.

Queria também agradecer a todos os membros da *Muvu Technologies*, que me acolheram de braços abertos. Em especial ao António Gonçalves, Alexander Fernandes, Dinis Faustino, Alexandre Emídio e Paulo Ilhéu, que em diferentes pontos ajudaram no desenvolvimento desta dissertação de tese.

Agradecer à minha Mãe, restante família e aos meus amigos pelo apoio, não só ao longo desta dissertação de tese, mas também durante todo o meu percurso.

Aos meus avós, que já partiram há algum tempo, quero agradecer pelo apoio que sempre demonstraram e terem em parte feito de mim quem sou hoje, especialmente o meu Avô Raposo, que foi um grande amigo e tanto me ensinou e incentivou a ser melhor.

Por fim, quero deixar um agradecimento muito especial ao meu Pai, que infelizmente já faleceu e não esteve presente durante a realização desta dissertação de tese, mas há de estar sempre presente no meu pensamento. Pela amizade, por todos os ensinamentos e lições, pelo apoio demonstrado ao longo dos anos, por isto tudo e muito mais que levarei para o resto da vida e não consigo expressar por meras palavras, o meu profundo e sincero agradecimento a ele. Obrigado Pai, sei que onde quer que estejas hás de estar orgulhoso do teu filho.

«You should never think it's easy.
It doesn't matter how good you are.»
(Mark Knopfler)

Resumo

A evolução da tecnologia ao longo dos anos tem tido um grande impacto em diferentes áreas. A indústria é uma das áreas que tem evoluído e beneficiado da implementação de soluções tecnológicas, que visam automatizar e monitorizar os processos industriais que ocorrem a nível de chão de fábrica, aumentando os níveis e qualidade de produção. Esta nova revolução industrial tem sido apelidada de Indústria 4.0 e é caracterizada por uma forte aposta em soluções tecnológicas.

Uma das vertentes de aplicação destas soluções tecnológicas da Indústria 4.0 é o planeamento industrial inteligente (*Smart Scheduling*). Este planeamento inteligente tem o objetivo de otimizar o processo de produção industrial, através de uma redução de custos de produção, cumprindo todos os requisitos de encomendas feitas ao fabricante. Como este problema se trata de um problema de otimização, a utilização de Algoritmos Genéticos para encontrar uma solução otimizada, pode ser uma das abordagens de resolução a adotar.

Esta dissertação de tese tem o objetivo de desenvolver uma solução capaz de gerar planeamentos de produção, que garantam uma redução de custos de acordo com as necessidades do fabricante, recorrendo a Algoritmos Genéticos, e através dos resultados obtidos analisar a viabilidade desta abordagem ao problema. Este projeto foi desenvolvido em parceria com a *Muvu Technologies*, que é uma empresa focada no desenvolvimento de soluções tecnológicas para a área industrial.

Nesta dissertação é feito um enquadramento geral do problema, apresentadas quais as abordagens propostas para solucionar o mesmo, bem como ferramentas e trabalho relacionado realizado e publicado na área, seguindo-se do desenvolvimento e implementação da solução, testes, validação em ambiente real e respetiva análise e conclusões.

Palavras-chave: Algoritmo Genético, *Muvu Technologies*, Indústria 4.0, *Smart Scheduling*

Abstract

The evolution of technology over the years has had a major impact on different areas. Industry is one of the areas that has evolved and benefited from the implementation of technological solutions, which aim to automate and monitor industrial processes that occur on the factory shop floor, increasing production levels and quality. This new industrial revolution has been dubbed Industry 4.0, and is characterized by a strong focus on technological solutions.

One of the areas of application of these Industry 4.0 technological solutions is *Smart Scheduling*. This *Smart Scheduling* aims to optimize the industrial production process, through a reduction in production costs, fulfilling all order requirements placed with the manufacturer. Since this problem is an optimization problem, the use of Genetic Algorithms to find an optimized solution could be one of the approaches to adopt for resolution.

This thesis dissertation aims to develop a solution capable of generating production schedules that ensure a cost reduction in accordance with the manufacturer's needs, using Genetic Algorithms, and through the results obtained, analyze the feasibility of this approach to the problem. This project was developed in partnership with *Muvu Technologies*, which is a company focused on developing technological solutions for the industrial area.

This dissertation presents a general framework of the problem, presenting the proposed approach to solving it, as well as tools and related work carried out and published in the area, followed by the development and implementation of the solution, testing, validation in a real environment and respective analysis and conclusions.

Keywords: Genetic Algorithm, Muvu Technologies, Industry 4.0, Smart Scheduling

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

1

Introdução

Neste capítulo é apresentada a motivação e contexto que levaram à realização deste projeto, seguido de uma exposição do problema, objetivos, a solução proposta para atingir os mesmos e as contribuições. Por fim, é apresentada a estrutura do documento.

1.1 Motivação e Contexto

Este projeto foi desenvolvido em conjunto com a empresa *Muvu Technologies* [41], que desenvolve soluções tecnológicas para a área industrial, com o objetivo de digitalizar e automatizar processos.

Figura 1.1: Muvu Technologies [41]

Nos últimos anos temos observado uma nova revolução industrial, denominada por Indústria 4.0, caracterizada pela aposta em soluções tecnológicas [51]. Todas as anteriores revoluções industriais ficaram marcadas por uma mudança de paradigma, sendo que na primeira foi o desenvolvimento de máquinas a vapor, na segunda o surgimento de sistemas elétricos e na terceira a automatização de processos recorrentes e posteriormente o surgimento de sistemas computacionais, mesmo que ainda primitivos [36, 42]. Prevê-se que cada vez seja mais comum a existência de *smart factories*, resultante da contínua evolução tecnológica e da vontade, por parte das empresas, em melhorarem os seus índices e qualidade de produção, de forma a manterem-se competitivas no mercado industrial.

Uma forte componente desta revolução industrial é a aposta em sistemas *MES* (*Manufacturing Execution System*). Este tipo de sistemas permite aumentar os níveis e qualidade de produção de uma fábrica de forma significativa, através da implementação de tecnologias e conceitos relacionados com a Indústria 4.0. O objetivo de um *MES* é digitalizar ao máximo os processos do dia-a-dia de uma fábrica, de forma a aproximar a mesma o máximo possível do conceito de *smart factory*. A informação disponibilizada por um

MES auxilia os responsáveis da fábrica, que estão ligados às diferentes componentes da produção e gestão. Esse conhecimento facilita o melhoramento contínuo da produção a diferentes níveis [27]. De modo geral, um sistema do tipo *MES* permite interligar a parte operacional do chão de fábrica com a gestão de topo da fábrica.

A *Muvu Technologies* desenvolve o sistema *RAILES*, que é um *MES* baseado nos princípios da Indústria 4.0. O *RAILES* permite analisar e controlar produções em tempo real e realizar a gestão de processos e recursos relacionados com a logística interna da fábrica. Hoje em dia, é cada vez mais evidente a aposta, por parte das empresas, na modernização e digitalização de processos e informação, através de sistemas como o *RAILES*, que permitem interligar o mundo físico com o digital [41].

A otimização do processo industrial, através da utilização de soluções tecnológicas, pode ser atingida em várias vertentes. A digitalização dos processos e da informação, juntamente com a análise em tempo real da mesma, é um dos caminhos que leva a esta otimização, pois leva a uma maior eficiência e qualidade na tomada de decisão. Esta digitalização depois abre caminho a outras soluções muito úteis na área industrial, como é a aplicação de técnicas de inteligência artificial, como aprendizagem automática e profunda, onde estas são utilizadas para, por exemplo, suportar soluções de qualidade preditiva, em que o objetivo é reduzir significativamente a quantidade de unidades defeituosas produzidas [50].

O planeamento de produção industrial de um fabricante é um aspeto decisivo no sucesso do mesmo. A necessidade de satisfazer as encomendas dos clientes, nos prazos delimitados e garantindo a qualidade do produto, é um dos desafios diários no mundo industrial. Para além disto, no final de todo este processo de produção existem muitos fatores que impactam os custos de produção e que dependem do planeamento executado. A redução de custos de produção é algo muito valorizado pelos fabricantes e por isso soluções que permitam atingir este objetivo são muito procuradas hoje em dia. Um planeamento cuidado e otimizado é uma das formas de reduzir os custos de produção, pois tem um impacto direto em muitos fatores que compõem esses custos. A acrescentar a este fator também podemos referir o avanço que um planeamento otimizado pode ter a nível de impacto ambiental e alterações climáticas, na medida em que pode permitir obter uma redução de consumo energético e de unidades produzidas descartadas.

A questão coloca-se na dificuldade em muitas vezes construir um planeamento otimizado devido à complexidade do problema e é aí que o recurso a uma ferramenta tecnológica de planeamento inteligente, incorporada num *MES* como o *Railes*, pode ser altamente vantajosa.

Este projeto vem no seguimento do desenvolvimento desta ferramenta de planeamento industrial inteligente, levada a cabo pela *Muvu Technologies*. O desenvolvimento enquadra-se no âmbito do projeto *"AI4MOS - Artificial Intelligence-based Multi-Objective Scheduling Optimization for Sustainable Manufacturing"*, submetido pela *Muvu Technologies* no contexto do projeto europeu *"KITT4SME platform-enabled KITs of arTificial intelligence FOR an easy*

uptake by SMEs" [30], que é suportado pelo programa de financiamento por fundos europeus *"Horizon 2020 - The EU Framework Programme for Research and Innovation"*, [23]. O projeto *AI4MOS* da *Muvu Technologies* visa explorar a extração de dados das máquinas integradas no sistema *RAILES* e utilizar então esses dados para criar modelos de Inteligência Artificial capazes de otimizar o planeamento industrial, seguindo a arquitetura de referência *KITT4SME* e alguns componentes já disponíveis, como ferramentas para recolha de dados ou o desenvolvimento de novos modelos. Posto isto, o objetivo é desenvolver e implementar uma solução de planeamento industrial inteligente, recorrendo a Algoritmos Genéticos, para a área de produção de plásticos, através do processo de moldagem por injeção. A *Muvu Technologies* possui uma empresa parceira neste projeto, que é cliente e tem o *RAILES* incorporado na sua fábrica. Esta empresa trabalha na área de moldagem por injeção de plástico.

A abordagem passou por analisar e modelar o problema de planeamento industrial, desenvolver e implementar uma solução, recorrendo a algoritmos genéticos, e por fim integrar a solução e testar o mesmo em diversos cenários. Aquando do início do desenvolvimento da tese, este projeto já se encontrava em desenvolvimento, no entanto o trabalho desenvolvido foi no seguimento do trabalho realizado anteriormente, dando a continuidade devida e tendo por base alcançar os objetivos do mesmo.

1.2 Exposição do Problema

O problema de planeamento da produção é bastante comum na área industrial, ou seja, a tarefa de processar várias encomendas da melhor forma possível, em que cada encomenda tem diferentes requisitos e prazos de entrega, sendo necessário satisfazer o cliente, salvaguardando o maior lucro possível para o fabricante.

Pensando num exemplo simples com poucas encomendas, poucos produtos e muitas máquinas e operadores disponíveis, não é uma tarefa difícil definir um planeamento ótimo ou perto disso, em tempo útil. No entanto, os casos reais caracterizam-se por uma maior quantidade e dimensão das encomendas, onde o número de máquinas e operadores disponíveis nem sempre é o ideal, para além de muitas outras especificidades relacionadas com o processo industrial em si, como certas características das máquinas e produtos.

De modo a melhor expor o problema em causa, em seguida é apresentado um possível caso. Neste exemplo temos três máquinas (M1, M2 e M3) e três produtos diferentes (P1, P2 e P3). A máquina M1 pode produzir P1 e P2, a máquina M2 pode produzir P2 e P3 e a máquina M3 apenas pode produzir P2. Na Tabela 1.1 são apresentadas as características relacionadas entre máquinas e produtos, como o *scrap rate*, que é a taxa média de unidades defeituosas produzidas que não passam no controlo de qualidade, e o ciclo médio de produção por unidade. O *scrap rate* é um fator importante pois se a encomenda for de k unidades então é necessário produzir $k * (1 + Sr)$, onde Sr é o *scrap rate*. Não é possível prever com exatidão quantas unidades produzidas serão descartadas, por isso apenas

podemos realizar uma previsão tendo em conta os valores de *scrap rate* de produções anteriores. Na Tabela 1.2 é apresentado consumo energético médio de cada máquina.

Máquina	**Produto**	***Scrap Rate***	**Ciclo médio por unidade (segundos)**
M1	P1	0.03	32.1
M1	P2	0.05	29.9
M2	P2	0.02	32.3
M2	P3	0.06	17.9
M3	P2	0.04	30.6

Tabela 1.1: Exemplo de relação entre máquinas e produtos

Máquina	**Consumo Médio (kW/h)**
M1	98.3
M2	95.7
M3	97.6

Tabela 1.2: Exemplo de consumo médio (kW/h) por máquina.

Na Tabela 1.3 são apresentadas as várias encomendas das encomendas realizadas à fábrica. Neste exemplo são utilizadas cinco encomendas. Estas encomendas irão dar origem a várias produções, no contexto do *Railes*. Uma produção é o termo utilizado para referir o processo de produzir uma certa quantidade de unidades de um certo produto. Uma produção pode ser composta por várias operações, ou seja, em alguns casos é necessário realizar uma sequência de processos para obter uma unidade do produto final. No caso explorado neste projeto, cada produção apenas é composta por uma operação, que é o processo de injeção. Também é importante referir que num caso real, uma produção assim que é iniciada numa determinada máquina é levada até ao fim nessa mesma máquina, salvaguardando os casos em que ocorra algum imprevisto, como a avaria da máquina. Por exemplo, no contexto da injeção de plástico, é necessário produzir 8000 unidades de um determinado produto. À partida a solução ideal poderia passar por produzir 5000 numa máquina e as restantes 3000 unidades noutra no entanto, em termos práticos isto pode não ser viável, pois provavelmente seria necessário realizar uma troca de molde na máquina, que por norma é um processo que dura várias horas e que pára a produção nesse período de tempo. Este fator do tempo necessário para preparar a máquina, como é a troca de molde, deve ser um aspeto a ter em conta aquando da realização do planeamento.

Encomenda	**Produto**	**Quantidade**	**Data Limite**
001	P1	5000	20/03/2023 14:00:00
002	P2	3000	17/03/2023 18:00:00
003	P1	9000	03/04/2023 10:30:00
004	P3	1500	21/03/2023 16:00:00
005	P2	5200	10/03/2023 11:00:00

Tabela 1.3: Exemplo com produções

O objetivo agora seria construir um planeamento de produção ideal para cumprir os requisitos das encomendas e minimizar os custos de produção. À partida podemos pensar que o mais vantajoso é produzir um certo produto na máquina que tem menor ciclo por unidade para esse mesmo produto, pois assim acabaria a produção mais rapidamente e seria mais fácil cumprir o prazo de entrega. A questão é que o ciclo por unidade não pode ser a única característica a ter em conta. Por exemplo, a máquina M1 é a que possui menor ciclo por unidade para o produto P2, no entanto também é a que possui maior valor de *scrap rate*, ou seja, apesar de produzir cada unidade em menor tempo, em relação às outras máquinas, também iria precisar de produzir mais unidades, pois em princípio um maior número seria descartado. Para além disso, também é a máquina que possui o maior consumo energético, logo o seu funcionamento é a que implica um custo maior ao fabricante nesse aspeto, o que reduz a margem de lucro do mesmo, para não falar da preocupação com as questões ambientais que cada vez mais relevância têm na nossa sociedade.

Apenas com um exemplo simples, podemos ter a noção que este é um tipo de problema de otimização complexo, no entanto a sua resolução traz uma grande vantagem para os fabricantes e por isso é uma solução cada vez mais procurada na indústria.

1.3 Objetivos

O principal objetivo deste projeto é desenvolver e implementar um Algoritmo Genético que seja capaz de produzir um planeamento de produção industrial o mais otimizado possível, de acordo com alguns fatores e as necessidades do fabricante, de forma a averiguar se esta é uma abordagem viável para a resolução deste tipo de problemas num contexto real.

Numa primeira fase, o objetivo passou por entender bem o conceito de Algoritmo Genético, as suas características, casos reais onde tenha sido aplicado, como pode ser utilizado para solucionar o problema e em seguida identificar as tecnologias já existentes que podem fornecer o suporte necessário para realizar a sua implementação. Este foi o principal alvo de trabalho ainda na fase de preparação, visto que foi a base de trabalho para as fases que se seguiram.

O modelo desenvolvido deve refletir o problema de planeamento real numa fábrica, mais em concreto, na industria da injeção de plástico. Outro aspeto a ter em conta é desenvolver o modelo tentando torná-lo o mais flexível possível, na medida em que este deve ir de encontro às diferentes necessidades de diferentes fabricantes e não seja necessário refazer totalmente o mesmo, aquando da adição de novas funcionalidades e adaptações a outras áreas industriais.

O algoritmo deve ser capaz de apresentar bons resultados em tempo útil. Não é fácil definir o que é considerado tempo útil ou não para a ação de planear. O ato de planear a produção de uma fábrica é algo feito com alguma antecedência, o que leva a que exista alguma tolerância na obtenção de resultados, por parte do utilizador. Mesmo assim o

tempo de execução do algoritmo é um fator importante e que deve ser minimizado sempre que possível.

Também é importante que o algoritmo desenvolvido seja compatível com sistema *RAILES* da *Muvu Technologies*, na medida em que os dados necessários devem ser disponibilizados pela plataforma e que a solução vá de encontro às necessidades da mesma.

Depois de implementado e testado, deve ser possível aplicar o algoritmo a um caso real, analisar os resultados obtidos e tirar conclusões dos mesmos. Tendo em conta os resultados obtidos, será possível realizar um balanço acerca da viabilidade do uso de Algoritmos Genéticos neste tipo de problemas de planeamento de produção industrial.

1.4 Solução Proposta

O problema de planeamento apresentado pode ser resolvido recorrendo a um algoritmo genético e é esse o objetivo deste projeto. É importante que a solução desenvolvida, para além de satisfazer as necessidades já apresentadas também seja flexível, ou seja, deve ser possível facilmente adaptá-la a diferentes realidades industriais e que o próprio algoritmo seja capaz de lidar com situações adversas, que podem ocorrer num caso real.

O planeamento gerado pelo algoritmo deve tentar otimizar um ou mais aspetos, que vão de acordo com as necessidades do fabricante. Esta otimização pode expressar-se de diversas formas, mas no final sempre se traduz numa questão de redução de custos e aumento do lucro para o fabricante, pois é o principal fator que leva as empresas a apostar neste tipo de soluções.

Posto isto, a função objetivo deve expressar esta vertente, ou seja, tentar maximizar o lucro do fabricante. A questão que se coloca é que maximizar diretamente o lucro nem sempre é viável, pois este está dependente de variáveis como preços de venda, contratos com fornecedores, despesas, entre outras coisas. Isto iria aumentar significativamente o número de variáveis do modelo e estas variáveis podem possuir alguma especificidade relacionada com o contexto do negócio, o que tornaria ainda mais difícil a generalização do modelo para diferentes casos. Para além que nem todos os fabricantes podem estar disponíveis para fornecer este tipo de dados para integrar na plataforma, o que inviabiliza desde logo a solução.

No entanto, existem outras formas de indiretamente maximizar o lucro, através da análise de alguns fatores que contribuem para o mesmo, como é a quantidade de *scrap* (unidades descartáveis produzidas). Minimizar a quantidade de *scrap* leva a menos desperdício de matéria prima e redução no tempo de produção que não acrescenta valor à fábrica, logo faz sentido a inclusão deste fator na função objetivo.

O cenário comum para estes fabricantes é existirem muitas encomendas e ser difícil dar resposta a todas, por isso sempre que uma máquina termina uma produção e fica livre é sinónimo de que pode ser iniciada uma nova produção. Sendo assim, aumentar a disponibilidade de máquina é uma forma de aumentar o lucro. Mas aumentar a disponibilidade de máquina é equivalente a minimizar o tempo em produção.

Uma forma de minimizar os custos de produção pode, em parte, passar por minimizar o consumo energético proveniente das máquinas. Esta abordagem é viável a nível de integração com o sistema *RAILES*, pois este recolhe várias informações sobre cada máquina, incluindo o consumo energético. Para além que minimizar o consumo, também permite aliar mais facilmente ao modelo as preocupações ambientais já referidas e que cada vez mais são tidas em conta pelos fabricantes.

Um fator que tem grande impacto na disponibilidade de máquina é o tempo de *setup* de cada produção, que corresponde ao tempo necessário para preparar o início de uma produção. Muitos fabricantes têm a preocupação de reduzir ao máximo o tempo de *setup* nos seus planeamentos e por isso este é um fator bastante relevante e que deve ser tido em conta na função objetivo.

Neste tipo de problemas é importante definir as variáveis e constantes do problema, tendo em conta o objetivo e contexto do mesmo. Neste caso podemos identificar logo duas componentes fulcrais no problema, que são os produtos e as máquinas, sabendo que nem todas as máquinas podem produzir o mesmo tipo de produto que outras. Posto isto, as principais variáveis do problema serão as que definem se um certo produto é alocado numa certa máquina e qual a ordem e quantidade de produção. Os valores destas variáveis são os "genes"que constituem os "cromossomas", que por sua vez representam as soluções durante a execução do algoritmo (tema abordado no Capítulo 2). Depois existem as constantes do problema, que de forma geral estão ligadas com as exigências das encomendas (ex: quantidades totais e prazo de entrega) e características das máquinas e produtos. As características das máquinas são por exemplo o consumo médio e tempo de ciclo da máquina para produzir uma unidade de um certo produto, que são recolhidos pelo *RAILES*. Também existe a questão do *scrap rate* de uma certa máquina com um certo produto, que é a taxa média de unidades produzidas que são descartadas. Depois temos as constantes em relação aos produtos das encomendas que são as quantidades e prazos de entrega, para além de questões relacionadas com a compatibilidade de certos produtos com certas máquinas, ou caso se aplique, a quantidade de moldes disponíveis e restrições de tempo causadas por possíveis trocas de molde nas máquinas.

A nível de restrições, estas servirão para garantir que nas soluções geradas são cumpridas as necessidades das encomendas, bem como numa situação real, o planeamento é viável a nível logístico. Em relação aos prazos de entrega, é mais vantajoso colocar essa componente como um fator penalizador na função em objetivo, pois passa a ser possível diferenciar diferentes graus de incumprimento e também pode acontecer que não seja possível cumprir os prazos e nestes casos é suposto que o algoritmo consiga gerar um planeamento, que mesmo não cumprindo o prazo continua a ser o melhor possível naquela situação.

Sendo assim, o *output* do algoritmo deve permitir construir um plano, em que produções são alocadas temporalmente às diferentes máquinas.

1.5 Contribuições

As contribuições esperadas levadas a cabo com a realização deste projeto são as seguintes:

- Desenvolvimento de uma solução para o problema de planeamento de produção industrial, recorrendo a Algoritmos Genéticos;
- A solução desenvolvida é capaz de produzir planeamentos que vão de encontro às necessidades do fabricante;
- Compatibilidade da solução desenvolvida com o sistema *RAILES* da *Muvu Technologies*;
- Validação da solução em ambiente real industrial e análise de resultados;
- Conclusões acerca da viabilidade do uso de Algoritmos Genéticos neste tipo de problemas de planeamento de produção industrial.

1.6 Organização do Documento

Este documento está dividido em cinco Capítulos, que possuem o intuito de expor o trabalho realizado neste projeto de dissertação.

Neste Capítulo de Introdução foi exposta a motivação e contexto do problema, junto com uma análise e exposição do mesmo. Também foram identificados os objetivos da realização deste projeto, bem como a solução proposta e as contribuições esperadas.

No Capítulo 2 é apresentado um Estado da Arte sobre as tecnologias e conceitos relacionadas com o projeto, possíveis ferramentas e trabalhos publicados na área.

No Capítulo 3 é descrito o processo de desenvolvimento da solução, através da apresentação dos componentes da mesma e justificando as principais decisões tomadas na implementação.

Em seguida no Capítulo 4 é descrita a metodologia de testes realizada em ambiente simulado e real, seguida da apresentação dos resultados obtidos e respetiva análise.

Por fim, são apresentadas as conclusões relativas ao trabalho realizado e as perspetivas de trabalho futuro a realizar.

2

Estado da Arte

Neste capítulo é apresentado um estado da arte em relação aos temas abordados neste projeto. Em primeiro lugar é abordado o tema de Algoritmos Genéticos, onde é realizado um enquadramento da tecnologia e como funciona este método, seguido da apresentação das principais ferramentas relacionadas existentes e por fim é feita uma análise de trabalhos publicados e relacionados com o tema. Em segundo lugar é feita uma análise do problema de planeamento industrial, apresentando as diferentes vertentes do problema, quais são as principais abordagens existentes para resolver o mesmo e trabalhos relacionados publicados.

2.1 Algoritmos Genéticos

Como o objetivo deste projeto é desenvolver um algoritmo genético para solucionar o problema de planeamento de produção industrial, então a principal tecnologia a ser utilizada são os algoritmos genéticos. Nesta secção é abordado o tema de algoritmos genéticos, através de um enquadramento geral inicial, seguida da descrição do funcionamento dos mesmos.

2.1.1 Enquadramento Geral

Um algoritmo genético é um método de pesquisa meta-heurístico e um tipo de algoritmo evolutivo inicialmente proposto por *J.H. Holland* em 1992 [22]. Hoje em dia são bastante utilizados para resolver problemas de otimização, como problemas de planeamento ou roteamento [31, 29].

Um problema de otimização possui um conjunto de variáveis e uma função objetivo a ser maximizada ou minimizada. Este tipo de problemas também pode possuir um conjunto de restrições, que acabam por restringir o conjunto de soluções possíveis para o problema. A complexidade de resolução para problemas de otimização está então relacionada com estes três fatores. A partir de um certo nível de complexidade, é necessário recorrer a métodos de resolução que nos garantam soluções em tempo útil, pois torna-se inviável resolver este tipo de problemas por meio de métodos heurísticos mais simples.

Os algoritmos genéticos não garantem que seja sempre descoberta a solução ótima do problema, no entanto permitem obter boas soluções, de forma consistente e em tempo útil, podendo estas, no melhor dos casos, corresponderem a soluções ótimas.

Os algoritmos genéticos são inspirados em conceitos da evolução biológica das espécies [39, 31, 29]. Estes baseiam-se no fenómeno de seleção natural, que é a base da Teoria da Evolução apresentada em 1859 no livro "A origem das espécies", por Charles Darwin, e num processo biológico denominado meiose.

A meiose é o processo de divisão celular de onde resultam as células sexuais dos seres vivos de reprodução sexuada. Neste processo, ocorre um fenómeno denominado *"crossover"*, onde diferentes cromossomas trocam porções de genes entre si, levando assim, a que os códigos genéticos das células resultantes da divisão sejam diferentes entre si e do código da célula original. Este fenómeno e por vezes mutações nos genes, que ocorrem de forma aleatória, levam à alteração de certas características, o que propulsiona a evolução das espécies [39, 55].

O conceito de seleção natural baseia-se na ideia que características favoráveis num certo ambiente irão tornar-se mais comuns ao longo de gerações da população desse ser vivo, e características menos favoráveis tornar-se-ão menos comuns. Isto leva a que exista uma evolução das espécies, de modo a melhor adaptarem-se a um certo ambiente, pois as características mais favoráveis prevalecem para as gerações seguintes [24].

Um algoritmo genético tenta adaptar esta teoria da evolução e o conceito de meiose, de modo a encontrar a melhor solução possível para um certo problema de otimização [26, 31, 29, 39].

2.1.2 Metodologia

A metodologia de um algoritmo genético pode ser dividida em várias fases e cada uma destas fases desempenha um papel fundamental. Em seguida são apresentadas e descritas cada uma destas fases.

2.1.2.1 Inicialização

Inicialmente é necessário gerar uma população de soluções. Esta população é composta por soluções que são geradas de forma aleatória. O tamanho da população está dependente do teor do problema, mas o ideal é ter uma população onde exista uma grande variabilidade de soluções que consiga representar o espaço de soluções possíveis. Em certos casos, podemos focar a população em zonas específicas do espaço de soluções, onde é mais provável ser encontrada a solução ótima [31].

Quanto maior a variabilidade de soluções na população inicial, menos provável o algoritmo cair num caso de máximo ou mínimo local, que pode não representar a melhor solução para o problema.

Cada solução é referida como sendo um "cromossoma"e os "genes"são porções desses "cromossomas"que representam as variáveis do problema [31, 29]. Na Figura 2.1 podemos

observar um exemplo de uma representação de uma população de cromossomas e os seus genes.

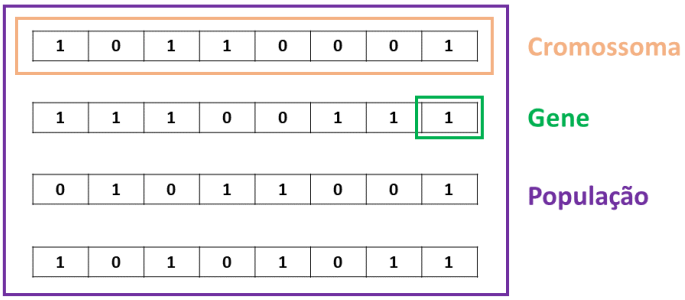

Figura 2.1: Cromossoma, gene e população.

2.1.2.2 Processo de Seleção

Depois de gerada a população inicial, cada uma das soluções é então avaliada de acordo com a função objetivo. Em seguida, é levado a cabo um processo de seleção onde são escolhidas as soluções que serão mantidas para a fase de reprodução. Este processo de seleção é repetido a cada iteração na execução do algoritmo.

Existem várias formas de executar o processo de seleção, como o método da roleta, onde a cada solução é selecionada de acordo com um valor de probabilidade atribuído. Este valor de probabilidade é atribuído consoante o valor da solução na função objetivo, ou seja, quanto melhor for a solução mais provável que esta seja selecionada. Também existe o método de torneio, onde soluções são colocadas frente a frente, de forma aleatória, e a melhor é selecionada [28, 29]. Por vezes, também pode fazer sentido selecionar automaticamente um pequeno conjunto de melhores soluções da população, de modo a não correrrmos o risco de estas não serem selecionadas. Este conjunto de soluções é denominado Elite. As soluções do conjunto de Elite podem apenas ser selecionados para a fase de reprodução ou também integrarem a próxima geração.

Um fator importante é ser possível manter alguma aleatoriedade no processo de seleção, mesmo que isto signifique selecionar piores soluções. Isto porque, é importante manter uma diversidade de soluções na população e evitar o problema de em poucas iterações o algoritmo convergir para uma solução que represente um máximo ou mínimo local da função objetivo e não esteja perto de ser a solução ótima para o problema [31].

2.1.2.3 Reprodução

Depois de selecionadas as soluções, estas serão utilizadas para gerar uma nova população de soluções, por meio de operações de *"crossover"* e mutação de genes.

Na operação de *"crossover"* é realizada uma troca de genes entre dois cromossomas de soluções selecionadas, de forma semelhante ao processo biológico que dá pelo mesmo nome. Estas trocas de genes entre cromossomas resultará na criação de novos cromossomas

que representam soluções diferentes das anteriores que lhes deram origem [31, 3, 26]. Uma questão relevante neste processo é definir que genes é que são sujeitos a estas trocas, ou seja, qual zona dos cromossomas é utilizada nas trocas. Os pontos que delimitam essa zona são apelidados de *"cutting points"* ou pontos de corte, e por norma são ajustados à medida que é testado o algoritmo, de forma a tentar melhorar os resultados obtidos. Na Figura 2.2 é exemplificado uma operação de *crossover*.

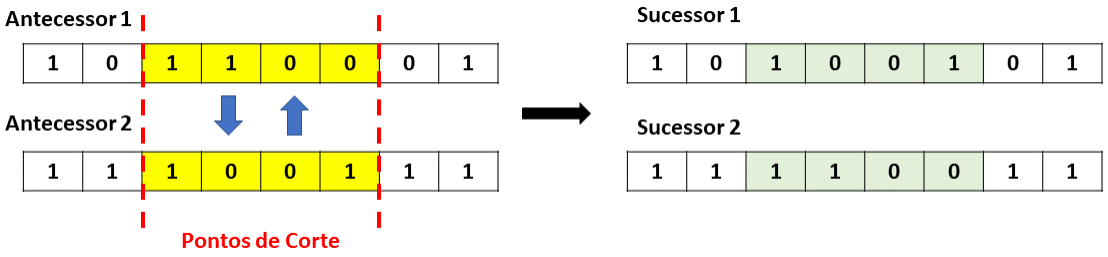

Figura 2.2: Operação de *Crossover*.

Estas novas soluções, resultantes da operação de *"crossover"*, ainda podem ser sujeitas a um processo de mutações aleatórias, para aumentar a diversidade das soluções e também diminuir um possível impacto negativo que o conjunto inicial gerado tenha no resto do processo. Uma mutação consiste em alterar o valor de um gene. Por exemplo, se tivermos genes representados em binário, como nos exemplos anteriores, então a mutação consiste em alterar valores de 0 para 1 e vice-versa. As mutações, por norma, ocorrem de acordo com um certo valor de probabilidade pré-definido, que deve ser ajustado à medida que é testado o algoritmo [31, 3, 26].

2.1.2.4 Término da Execução

Depois da inicialização, os processos de seleção e reprodução são repetidos iterativamente, até ser satisfeita uma certa condição de paragem. Existem várias condições de paragem que podem ser aplicadas, como não existirem melhorias em soluções de gerações consecutivas ou um limite no tempo de execução ou número de gerações [31, 3, 26]. No final, a melhor solução encontrada é a solução retornada pelo algoritmo.

2.1.3 Ferramentas Existentes

Existem ferramentas de suporte para Algoritmos Genéticos em diversas linguagens de programação, de modo a facilitar a implementação deste tipo de soluções. Neste projeto, a implementação será realizada na linguagem de programação *Python*, pois foi um requisito da *Muvu Technologies*, devido à compatibilidade com o projeto *AI4MOS*, e por isso a pesquisa realizada foi direcionada neste sentido.

O *Python* é uma linguagem de programação bastante utilizada na área da inteligência artificial, devido às diversas bibliotecas e *frameworks* muito úteis que suportam o desenvolvimento nesta área, como o *TensorFlow*, *Keras*, *Scikit-learn* ou *PyTorch*, aliadas a bibliotecas

como o *NumPy* e o *MatPlotLib*, para realizar operações matemáticas e desenvolvimento de gráficos, respetivamente.

No contexto de desenvolvimento de algoritmos genéticos, existem várias bibliotecas que podem ser úteis, como o *PyGAD* [15], *Pyevolve* [44], *LEAP* [9], entre outras. Apesar de serem diferentes, de modo geral, estas bibliotecas permitem executar um Algoritmo Genético, de forma fácil e rápida, através da simples definição de alguns processos e parâmetros. A nível de parâmetros é possível definir vários fatores como o número de gerações, número de genes que compõe o cromossoma, intervalo de valores para cada gene, tamanho da população, numero de indivíduos selecionados a cada geração, entre outros. Quanto aos processos relacionados com a metodologia de execução do algoritmo, é possível definir alguns parâmetros, entre os principais: escolher o método de seleção, *crossover* ou mutação de um conjunto de métodos disponibilizados, a porção de elementos que compõem a elite e a probabilidade de ocorrer uma mutação.

Nas Listagens 1 e 2 são apresentados dois exemplos para a execução de um Algoritmo Genético para um problema simples, utilizando as bibliotecas *PyGAD* e *LEAP*, respetivamente.

```
import pygad
from pygad import selectors, crossovers, mutations

#Função Objetivo
def fitness_func(solution):
    return sum(solution)

#Parâmetros
num_generations = 100
num_parents_mating = 10
sol_per_pop = 20
num_genes = 10
mutation_prob = 0.01

ga_instance = pygad.GA(num_generations=num_generations,
                       num_parents_mating=num_parents_mating,
                       fitness_func=fitness_func,
                       sol_per_pop=sol_per_pop,
                       num_genes=num_genes,
                       parent_selection_type=selectors.Tournament,
                       crossover_type=crossovers.TwoPointCrossover,
                       mutation_type=mutations.BitFlip,
                       mutation_percent_genes=mutation_prob,
                       elitism=True)

#Executar o algoritmo
ga_instance.run()

#Melhor solução
solution, solution_fitness, solution_idx = ga_instance.best_solution()
```

Listing 1: Exemplo com a biblioteca PyGAD.

```python
from leap import Individual, EvolutionaryAlgorithm, Selection, Crossover, Mutation

#Função Objetivo
def fitness_func(chromosome):
    return sum(chromosome)

#Parâmetros
ea = EvolutionaryAlgorithm(fitness_func)
ea.population_size = 20
ea.num_generations = 100
ea.chromosome_length = 10
ea.selection = Selection.tournament_selection
ea.crossover = Crossover.two_point_crossover
ea.mutation = Mutation.bit_flip_mutation
ea.elite = True
ea.mutation_probability = 0.01

#Executar o algoritmo
ea.run()

#Melhor solução
best_individual = ea.best_individual
```

Listing 2: Exemplo com a biblioteca LEAP.

Mesmo assim, estas bibliotecas podem não ser suficientes, pois em problemas mais complexos e com um certo grau de especificidade, estas soluções generalizadas e simplificadas nem sempre conseguem responder às necessidades do problema. Por isso, muitas vezes é necessário realizar um desenvolvimento de raiz do algoritmo, de modo a este ir de encontro às necessidades do problema.

Devido a todas as especificidades presentes no problema de planeamento de produção industrial, desde o processo de gerar soluções, validar e avaliar as mesmas, é necessário realizar um desenvolvimento de raiz do algoritmo. Este tipo de bibliotecas, mesmo com todos os parâmetros e funcionalidades que permitem personalizar, não oferecem o controlo necessário, sobre os processos, para garantir uma implementação capaz de solucionar, de forma viável, o problema de planeamento de produção industrial.

2.1.4 Trabalhos Relacionados

Hoje em dia, existem vários tipos de problemas que são resolvidos por meio de Algoritmos Genéticos. Estes são por norma problemas de otimização complexos, caracterizados por um grande número de variáveis e restrições. O desenvolvimento destas soluções em diversas áreas, por norma, leva à publicação de vários trabalhos sobre as mesmas.

Assim, é importante realizar uma análise dos trabalhos relacionados com a implementação de Algoritmos Genéticos em diversas áreas, de modo a adquirir conhecimento que possa ser útil na implementação da solução para o problema de planeamento de produção industrial. Resultante da pesquisa e análise desses trabalhos, foi possível identificar algumas áreas onde os Algoritmos Genéticos são aplicados e explorar as soluções desenvolvidas envolvendo os mesmos.

Roteamento de Veículos

Um dos problemas clássicos de otimização, e que é bastante comum hoje em dia, é o problema do roteamento de veículos, conhecido por *VRP* (*Vehicle Routing Problem*), ligado à questão de logística, desde cadeias de distribuição ao planeamento de transportes públicos. Neste tipo de problemas, o objetivo é encontrar as melhores rotas, que cumprem as restrições impostas, minimizando os custos de entrega, através de fatores como o tempo de viagem, distância percorrida ou o número de veículos utilizados. Muitas vezes também é importante minimizar o tempo total de espera das encomendas ou pessoas nas suas estações, que este por sua vez equivale a minimizar a ocupação dos armazéns ou estações e a maximizar o volume de negócio [10, 45, 7].

A aliar a estas questões de minimizar custos e maximizar o volume de negócio nos problemas de roteamento, também é relevante a preocupação ambiental. Deste ponto de vista, cada vez são mais os esforços para reduzir a poluição atmosférica, causada estes veículos de transporte. Por isso esta também é umas das preocupações que leva à aposta em soluções, como os Algoritmos Genéticos, que permitam gerar planeamentos de rotas otimizados [13].

Neste tipo de problemas de roteamento, podem existir muitas variáveis e restrições. Isto torna-o num problema em que a complexidade escala muito rápido, de acordo com a dimensão da cadeia de distribuição ou de transporte e do grau de profundidade que queremos dar ao problema. Pensemos na tarefa que é organizar e planear o sistema de transportes públicos de autocarro de uma grande cidade, onde habitam milhares ou milhões de habitantes e em 2030 são esperadas que 5 mil milhões de pessoas habitem em áreas urbanas a nível global [45]. Para além da quantidade de variáveis (ex: autocarros, paragens, utilizadores, etc.) também existem as restrições impostas pelo contexto do problema, como por exemplo a ordem das paragens, a capacidade dos autocarros, os horários, etc.

Para um problema de roteamento, em que o objetivo é cobrir um conjunto de pontos de entrega através de rotas, que garantem a menor distância possível, pode ser utilizada uma heurística como, por exemplo, a de *Clarke and Wright*. Esta heurística baseia-se na construção das rotas necessárias de forma iterativa, através da junção de rotas intermédias que garantem a menor distância, aquando da junção [46]. Este tipo de abordagens permite obter soluções razoáveis em tempo útil, no entanto, não garante que solução seja ótima ou perto disso, pois o espaço de procura de soluções é limitado. Para além que heurísticas, como a de *Clarke and Wright*, são métodos pensados para problemas específicos e que podem não ser facilmente adaptáveis a outras variantes do mesmo problema.

Neste campo, os Algoritmos Genéticos ganham vantagem e acabam por ser uma das melhores formas de lidar com este tipo de problemas, pois permitem uma busca num espaço de soluções alargado e são adaptáveis a diferentes problemas, desde que a modelação do problema seja ajustada, visto que a metodologia base de resolução é idêntica para qualquer problema. Isto acaba por ser uma grande vantagem, pois deixa de

ser necessário desenvolver um método de resolução de raiz ou adaptar um já existente, para cada problema com características muito específicas, como por exemplo o "Freight Rail Transport Scheduling Problem" (FRTSP), onde são utilizadas as redes ferroviárias para fazer o transporte de mercadoria, de modo a ser uma solução mais ecológica e sustentável [45]. Este problema é ainda mais complexo do que o problema de transporte comum, pois existe um maior número de restrições e varáveis a ter em conta, que estão relacionados com o contexto dos transportes ferroviários (ex: horários que não são tão flexíveis como quando o transporte é realizado pela própria empresa; nem todos os sítios são diretamente acessíveis por rede ferroviária; etc.).

A nível de modelação de problemas de roteamento, são assumidos vários fatores, como os veículos e as suas características (ex: capacidade, consumo, velocidade, etc.), pontos de entrega, centros de distribuição, limitações nos horários de entrega, entre outros. Este fatores traduzem-se em restrições ou penalizações na função objetivo. Por exemplo, atrasos nos horários de entrega ou as cargas atribuídas a cada veículo, podem ser um fator penalizante na função objetivo a juntar ao custo associado ao processo de distribuição [10]. Quanto à variável de decisão de rotas, esta pode ser representada por uma matriz, onde cada linha corresponde à rota de um veículo e as entradas representam os pontos ordenados a serem visitados. Também é possível modelar a mesma situação através de uma matriz de alocação com valores binários, que representam se um certo veículo vai desde um certo ponto a outro, sendo que a rota total de um veículo é formada pela ligação sequencial destes trajetos entre dois pontos [10, 45]. Ambas são abordagens válidas, apenas implicam cuidados distintos, principalmente se a matriz for binária, pois é necessário ter mais restrições de verificação para garantir que esta é preenchida de forma correta.

Em relação aos horários de partida e chegada aos diferentes pontos das rotas, esses são calculados tendo por base a matriz da alocação ou já estão previamente definidos, como no caso do FRTSP [45]. Se for pretendido otimizar também os horários e não só a sequência de pontos das rotas, então o nível de complexidade aumenta e passa a ser necessário adaptar o modelo.

Quanto à execução de um Algoritmo Genético num problema de roteamento, esta segue a metodologia base já apresentada. É sempre possível realizar alguns ajustes e tomadas de decisão dentro dos processos tradicionais, como definir o método de seleção ou *crossover*, mas de modo geral o procedimento é mantido. O critério de paragem por norma são o número de gerações, tempo computacional ou não se verificar uma melhoria na solução encontrada durante várias gerações, no entanto também pode ser utilizado um número de gerações dinâmico como critério de paragem, como no caso do FRTSP, em que o número limite de gerações pode ser definido com base no número de comboios, estações e carga a transportar [45].

A variável de alocação é a principal constituinte do cromossoma. Os valores desta matriz são os genes utilizados nas trocas durante o *crossover* e mutação. Em vários casos, depois de gerar uma nova solução na fase de reprodução, é utilizado um algoritmo suplementar para melhorar essa solução gerada, através de uma procura local na vizinhança da

mesma. Esta procura é feita através de pequenas alterações nas rotas da solução original, verificando sempre se estas traduzem-se em melhoria da solução [10]. O processo de procura local pode ser aplicado em vários tipos de problema e não só ao problema de roteamento, sendo que a diferença pode ocorrer no método de procura e na definição de vizinhança utilizados.

Assim, podemos concluir que os Algoritmos Genéticos representam um dos métodos utilizados para resolver diferentes tipos de problemas de roteamento e que alguns aspetos destas soluções analisadas podem ser úteis no desenvolvimento deste projeto, pois existem similaridades em alguns fatores que levam a que certas ideias e processos possam ser adaptados à solução desenvolvida para o problema de planeamento industrial.

Planeamento

Os problemas de planeamento são transversais a várias áreas da nossa sociedade. Um exemplo é o próprio problema do planeamento industrial abordado neste projeto. Depois existem outro tipo de problemas de planeamento comuns no nosso quotidiano. Todo o tipo de problema que se resume à alocação de meios ou pessoas a certas atividades ou locais, enquadra-se na categoria dos problemas de planeamento. Depois podem existir algumas vertentes, como por exemplo, quando acrescentamos o fator de datas e horários ao problema, em que este fator passa a também ser alvo de otimização.

Um problema de planeamento comum é a construção de horários escolares. Nestes casos é necessário alocar aulas de turmas a salas num certo horário, otimizando ao máximo a utilização dos diferentes recursos, tendo em conta o tamanho das turmas, capacidade e características das salas, professores, possíveis conflitos, etc. Este problema é considerado *NP-hard* e não pode ser solucionado em tempo útil utilizando um método determinístico, pois o problema rapidamente escala de complexidade com o aumento do número de turmas, aulas, professores e salas, aliado a todas as possíveis restrições características do contexto do problema [59, 49].

O planeamento de horários escolares, e até mesmo de outro tipo de horários, muitas vezes tem a vantagem de serem caracterizados por uma alocação em certos intervalos de tempo padronizados e não em valores contínuos, pois as aulas têm durações fixas, independentemente de quando e em que sala são realizadas. Assim, podemos olhar para o horário como tendo uma estrutura 3D, com as dimensões a representar dias, intervalos de tempo e salas, sendo que os valores da matriz correspondem às aulas [49]. A nível de modelação, uma abordagem é considerar a matriz de alocações com valores binários, em que o valor 1 representa que um professor leciona uma aula específica para uma turma em um determinado dia e horário em uma sala específica, garantindo que cada sala, turma e professor não são alocados a aulas em simultâneo. Quanto à função objetivo, pode não ser fácil quantificar o valor de uma solução. Um critério pode estar relacionado com o quão cedo são as aulas, ou seja, quanto mais cedo diariamente as aulas foram alocadas melhor é a solução. Isto é baseado na ideia que os estudantes têm mais facilidade em aprender nas

primeiras horas do dia e vão perdendo essa capacidade e o interesse ao longo do mesmo [49].

Em relação à implementação de um Algoritmo Genético para este tipo de problemas de planeamento de horários, um fator chave no sucesso do mesmo é minimizar a probabilidade de serem criados conflitos nas soluções através das operações de *crossover* e mutação. Tendo em conta que existem muitas restrições, é fácil que uma alteração leva a que o horário gerado deixe de ser viável. Por isso, é importante minimizar esses casos, de modo a evitar que o algoritmo perca tempo a gerar indivíduos que não representam soluções viáveis. Este aspeto também é aplicável ao problema de planeamento industrial, pois as alterações provocadas nos indivíduos, pelas operações de *crossover* ou mutação, podem levar a que restrições relacionadas com as quantidades produzidas deixem de ser cumpridas, inviabilizando assim a solução.

A nível de transporte aéreo, os Algoritmos Genéticos também podem ser uma solução para as companhias aéreas conseguirem gerar planeamentos de voos e alocação de recursos nos mesmos, tendo em conta que em certas alturas do ano a procura é mais elevada. Este problema deriva dos tradicional *Fleet Assignment Problem* (FAM), em que para cada voo, cuja a hora de partida já está previamente definida, é alocado o avião com melhores características para realizar o mesmo [48]. No entanto, esta é uma abordagem simples e não responde a todas as necessidades impostas, o que leva à necessidade de adaptar e acrescentar mais complexidade ao problema. Existe a necessidade de ter em conta mais fatores, como por exemplo questões de manutenção e recursos humanos, e o objetivo passa a ser obter os melhores horários para os voos, para além de apenas alocar os aviões aos mesmos [48, 1]. Num caso ainda mais complexo, pode ser necessário também fazer o roteamento dos voos e neste caso esta vertente do problema tomaria alguns contornos semelhantes aos problemas de roteamento já apresentados.

Os problemas de planeamento, mesmo que aplicados a diferentes áreas e com diferentes vertentes, possuem características comuns, no que toca à sua modelação e a desafios de implementação em Algoritmos Genéticos. A análise destas soluções em diferentes casos permite obter conhecimento útil, que pode ser adaptado ao problema de planeamento industrial e ajudar a atingir os objetivos deste projeto.

Machine & Deep Learning

Machine Learning é uma das áreas da Inteligência Artificial que consiste em treinar um modelo para realizar uma determinada tarefa, através de um certo conjunto de dados e ajuste de parâmetros para melhorar a performance na execução da mesma.

Deep Learning é o termo utilizado para nos referimos aos métodos de *Machine Learning* que utilizam múltiplas camadas de transformações não lineares. Estes modelos são inspirados no funcionamento do cérebro humano, pois tentam recriar as interações entre os neurónios existentes no mesmo, e são especialmente úteis quando existe uma grande quantidade de dados que podem ser utilizados no processo de treino. Nos últimos anos

começou-se a dar mais importância a este tipo de abordagem, devido à da melhoria a nível de hardware, o que resultou numa evolução em relação aos estudos feitos nesta área. O desenvolvimento de melhor hardware, como a criação de GPU's mais eficientes e acessíveis no mercado, tornou possível a aplicação de conceitos teóricos a nível prático, que antes não eram viáveis, devido às limitações computacionais [38, 4].

Os dados com que treinamos estes modelos são peça chave no sucesso dos mesmos. Por vezes, existem *datasets* com uma grande quantidade de *features*, mas nem todas são úteis para o problema que queremos resolver e apenas dificultam o treino do modelo. O processo de seleção destas *features* é bastante importante e os Algoritmos Genéticos podem ajudar a encontrar o conjunto de *features* que melhor se enquadra para um certo problema e permite obter melhores resultados [32, 6, 25, 61]

Existem vários tipos de modelos de redes neuronais profundas e dentro de cada tipo existem diferentes arquiteturas e abordagens, o que faz com que a tarefa de construir uma destas redes seja um tarefa complexa, que normalmente é feita de forma iterativa e através de um método de experimentação, observação de resultados e ajuste do modelo. Em relação aos tipos de redes neuronais utilizadas no *Deep Learning*, cada tipo tem uma aplicabilidade diferente e são adequadas para problemas diferentes. As *Convolutional Neural Networks* (CNNs) são bastante usadas em problemas que utilizem imagens ou áudio, pois estas permitem identificar padrões e conseguem obter bons resultados com este tipo de dados. Também existem as redes de *Autoencoders*, que tentam reproduzir um certo input através de uma representação reduzida do mesmo. Estes podem ser utilizados em conjunto com as CNNs, por exemplo para segmentar partes de uma imagem [38, 4].

Um dos desafios das redes neuronais é definir a sua arquitetura, pois a sua performance ao realizar certa tarefa está bastante dependente da forma como são compostas as diferentes camadas do modelo. Os Algoritmos Genéticos podem ser utilizados para automatizar o design destas arquiteturas, com o objetivo de obter o menor erro possível. Para tal, é gerado um conjunto de arquiteturas a partir de componentes e camadas pré-definidas (onde, por exemplo no caso das CNNs, variam parâmetros como a dimensão do *kernel* de convolução, o número de filtros e o *stride*), que depois, através da metodologia dos Algoritmos Genéticos, será utilizado para tentar obter uma arquitetura que minimize o erro (função objetivo do problema), para um determinado *dataset* [52].

Estes modelos também são caracterizados muitas vezes pela quantidade de parâmetros que necessitam de ser ajustados manualmente e iterativamente, de modo a obter melhores resultados. O uso de Algoritmos Genéticos pode ser utilizado para realizar o ajuste dos valores destes parâmetros para certos problemas [35, 60, 54]

2.2 Planeamento Industrial

O problema de planeamento industrial é um dos grandes desafios que os fabricantes enfrentam. O planeamento de produção é um dos fatores deste planeamento industrial e é o principal alvo de estudo neste trabalho. Nesta secção é abordado o problema de planeamento industrial, as suas diferentes vertentes e possíveis métodos de solução para o mesmo.

2.2.1 Enquadramento Geral

O planeamento industrial de uma fábrica é um fator crucial no sucesso da mesma. A construção de um planeamento que minimize os custos e melhore os índices de produção é um dos grandes desafios que os fabricantes enfrentam, devido à elevada complexidade inerente da natureza do problema. Este planeamento pode ser dividido em várias vertentes, que compõem o planeamento industrial geral da fábrica, como podemos observar na Figura 2.3.

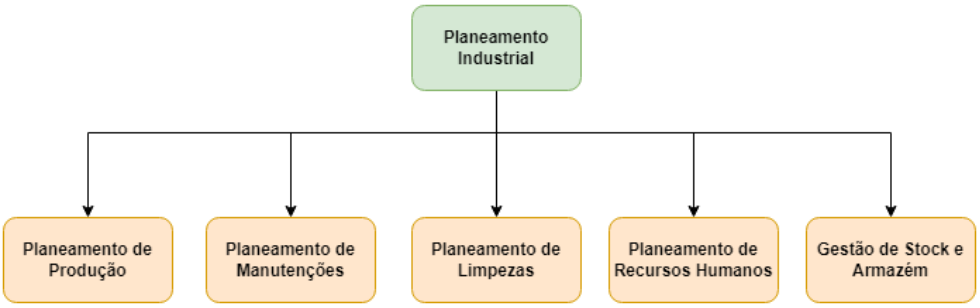

Figura 2.3: Decomposição do Planeamento Industrial.

A principal vertente abordado neste projeto é o planeamento de produção, que se baseia em organizar ao longo do tempo as operações de produção a realizar a nível de chão de fábrica. Mas este não é o único tipo de planeamento que ocorre a nível industrial. A acrescentar ao planeamento de produção é necessário realizar o planeamento de outros recursos e tarefas. O planeamento de recursos humanos ao longo dos diferentes turnos laborais é outra tarefa exigente e que possui impacto direto na produção e no seu planeamento, pois podem existir casos em que certas operações necessitam de operadores que possuam capacidades ou habilitações específicas.

Outro planeamento importante numa fábrica é o planeamento de manutenções e limpezas. As manutenções são intervenções realizadas a equipamentos e ferramentas, que podem ser compostas por uma ou mais tarefas. Existem as manutenções preventivas e as de recurso, que são causadas por avarias. Apenas as manutenções preventivas podem ser planeadas com antecedência e são planeadas tendo em conta as indicações dos construtores e fabricantes dos equipamentos. O critério utilizado para o planeamento é a periodicidade ou a carga de trabalho, ou seja, as manutenções de certos equipamentos são planeados tendo em conta um intervalo de tempo pré-definido, enquanto que as

manutenções de outros estão dependentes das horas de trabalho efetivo. Quanto às intervenções de limpeza, este é um processo bastante comum em indústrias alimentares ou em indústrias que trabalham com compostos químicos, como a indústria farmacêutica, devido à necessidade de higienizar ou descontaminar uma sala ou um equipamento.

Tanto o planeamento de manutenções como o de limpeza, estão dependentes do planeamento de recursos humanos, pois existem turnos para as equipas de manutenção e limpeza. Por outro lado, possuem impacto direto no planeamento de produção, visto que enquanto uma máquina ou um centro de trabalho está em manutenção/limpeza não pode estar a realizar nenhum tipo de operação.

A gestão de armazém e *stock* é outro tipo de planeamento que está naturalmente ligado com o planeamento de produção. Em muitas áreas indústrias, grande parte das operações necessitam de matéria prima para a realização da mesmas e é importante garantir que a quantidade necessário está disponível para realizar a operação. Por exemplo, é inviável planear uma produção para um certo intervalo de tempo se não tivermos a matéria prima necessária nessa altura. O mesmo se aplica para os planeamento de manutenção e limpeza, pois estes também necessitam de recursos para realizar as tarefas. Posto isto, é importante que os diferentes planeamentos e a gestão de armazém e *stock* estejam em sintonia e complementem-se, de forma a evitar quebras no funcionamento devido a mau planeamento à priori.

De um ponto vista ideal, o planeamento industrial tem em conta todos estes componentes e é otimizado num todo, pois o bom funcionamento da fábrica está dependente de cada um individualmente e da harmonia entre os mesmos. No entanto, não são muitos os casos reais onde isto ocorre. Por norma, o que acontece é que, os planeamentos de produção e manutenção são feitos de forma separada, tendo em conta o planeamento de turnos, e depois são ajustados entre si, muitas vezes em cima do acontecimento e conforme as necessidades. Esta metodologia de planeamento torna difícil a otimização do mesmo.

Neste projeto, o foco é exclusivamente o desenvolvimento de uma solução para o planeamento de produção. Contudo, é necessário que os outros aspetos do planeamento sejam considerados durante a criação do planeamento de produção. Em outras palavras, não é pretendido otimizar o planeamento de turnos, manutenções e limpezas. Em vez disso, o objetivo é otimizar o planeamento da produção, levando em consideração os planeamentos já estabelecidos para esses outros elementos.

2.2.2 Planeamento de Produção e Extensões

O problema de planeamento de produção industrial também é referido como *Job Shop Scheduling Problem* (JSSP), que representa a tarefa de determinar a ordem de execução de um conjunto de operações em máquinas, sendo depois possível, transformar esta ordem numa alocação temporal das operações [47]. Uma fábrica é composta por diferentes componentes que intervém diretamente no processo de produção. Dependendo da área industrial, a metodologia de produção pode variar, mas de modo geral uma fábrica é

composta por um conjunto de máquinas/equipamentos que realizam certas operações. Desta forma num JSSP o objetivo é distribuir da melhor forma um conjunto de operações pelas diferentes máquinas [58, 47]. Esta abordagem assume que cada operação apenas pode ser alocada a uma máquina específica. No entanto, num caso real, isto não se aplica pois diferentes máquinas podem realizar o mesmo tipo de operação, e associado a esta característica, existem diversos fatores que aumentam a complexidade do problema. A distribuição das operação pelas diversas máquinas está assim dependente destes fatores, que dependem muito da área industrial onde se aplica o problema, pois estão diretamente ligados aos processos e recursos envolvidos.

Este aumento da complexidade do problema leva-nos ao *Flexible Job Shop Scheduling Problem* (FJSSP), uma extensão do JSSP, que lida com maior flexibilidade na alocação de operações às máquinas, incluindo a possibilidade de usar diferentes máquinas para o mesmo tipo de operação [12, 33]. O FJSSP é uma representação mais realista do problema de planeamento de produção e dá melhor respostas às necessidades impostas na resolução do mesmo, sendo que é necessário adaptá-lo às especificidades do problema que estamos a abordar.

Em muitas industrias, um processo de produção para um certo produto é repartido por um conjunto de operações. Estas operações podem ser independentes entre si ou pode existir uma ordem de operações a cumprir na produção, como no exemplo ilustrado na Figura 2.4. A existência de precedência de operações na produção é um fator que aumenta a complexidade do planeamento, visto que é necessário garantir que a ordem é cumprida, caso contrário o planeamento não é exequível [12, 58, 47]. No caso da produção por molde e injeção de plástico, este cenário não se aplica, pois a produção de um produto é composta por apenas uma operação, que é a operação de injeção.

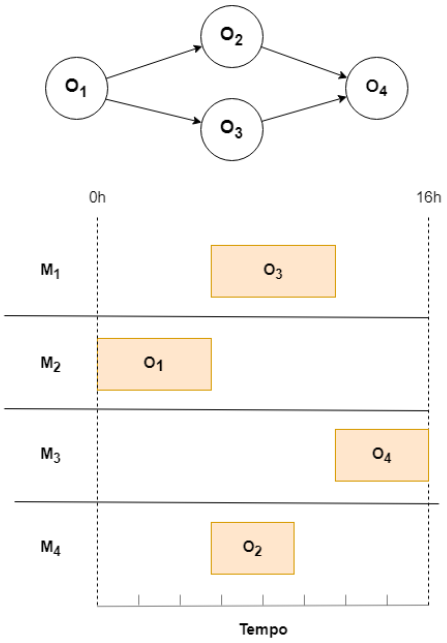

Figura 2.4: Exemplo de planeamento de uma produção com sequência de operações.

Para cada operação pode existir um tempo de *setup*, que equivale ao tempo necessário para preparar a realização da operação, e este pode variar entre máquina. Também pode acontecer que o tempo de *setup* varie dependendo do que foi produzido antes. Por exemplo, o tempo necessário para mudar da produção do produto 1 para o produto 3 pode ser diferente do tempo necessário para mudar do produto 2 para o produto 3. Dependendo do tipo de operação, este tempo de *setup* pode ser significativo e ter um impacto negativo no planeamento.

A nível de máquinas, cada uma possui as suas próprias características e a mesma operação realizada em máquinas diferentes pode ter um impacto diferente na produção, devido ao tempo, custo e nível de qualidade associados à máquina para aquela operação específica. Para cada operação, a máquina tem um tempo de ciclo de produção médio, que equivale ao tempo necessário para produzir uma tiragem, e um valor médio de *scrap rate*, que é a porção esperada de unidades produzidas com defeito. O tempo de ciclo por unidade está dependente do processo em si e ferramentas utilizadas no mesmo. Pensemos no problema da injeção de plástico, em que são utilizados moldes na produção. Cada molde pode possuir um número de cavidades diferente, em que cada cavidade equivale a uma unidade produzida. Neste caso, o tempo de ciclo por unidade é dado pelo tempo por tiragem a dividir pelo número de cavidades do molde.

O tempo de duração de uma operação pode ser fixo ou calculado através da quantidade produzida e do ciclo de produção por unidade. Caso a operação tenha um *setup*, então o tempo do mesmo também deve constituir duração da operação. A duração de uma produção é a soma das durações das operações que compõem a produção. Esta duração em produção ativa pode não corresponder à janela temporal da produção, pois as produções só estão ativas quando a fábrica está em funcionamento. Por exemplo, imaginemos uma fábrica que opera com dois turnos de produção diários, com duração de 8 horas cada, das 8:00 às 16:00 e das 16:00 às 0:00, com uma pausa entre as 0:00 e as 8:00. Nesta fábrica, qualquer produção que ultrapasse as 16 horas tem de ser obrigatoriamente interrompida pelo menos uma vez, devido à ausência de um turno de produção durante esse período. As pausas na produção também podem ser causadas por intervenções de manutenção ou limpeza.

Outro aspeto relevante ao realizar o planeamento de produção, está relacionado com a concorrência de operações do mesmo tipo, ou seja, quantas operações do mesmo tipo posso realizar em simultâneo. Este número está dependente das características de produção e da fábrica, em que no limite, este é igual ao número máximo de máquinas disponíveis. Se uma operação específica requer o uso de uma determinada ferramenta (ex: molde) e a fábrica possui apenas duas, então o número de vezes que essa operação pode ser realizada em simultâneo é 2.

As produções são geradas e planeadas de forma a colmatar a necessidade de satisfazer as encomendas realizadas ao fabricante. Uma encomenda é composta por uma data de entrega e a quantidade de um certo produto a entregar. O método mais simples, é cada encomenda gerar uma única produção que satisfaz a encomenda. Este acaba por ser o

método mais utilizado pelos fabricantes, pois simplifica a tarefa de construir o planeamento. No entanto, esta solução limita logo à partida o espaço de soluções para o problema, pois evita que uma encomenda seja dividida em várias produções, o que pode ser vantajoso em algumas situações. Por isso, no que toca a este ponto, existem duas vertentes para o problema: apenas otimização da alocação de produções com quantidades de produção já previamente definidas ou a otimização da alocação de produções e quantidades a produzir em cada produção. Esta segunda vertente leva a que o espaço de soluções a explorar seja maior, pois qualquer combinação entre produções e quantidades a produzir é válido, desde que cumpra as necessidades das encomendas. A maioria dos modelos e algoritmos apresentados em trabalhos publicados abordam a solução mais simples, que utiliza um número fixo de produções com quantidades previamente definidas. No entanto, quando consideramos a capacidade de lidar dinamicamente com a quantidade e produções, o problema torna-se mais complexo, mas essa abordagem possibilita encontrar soluções mais otimizadas.

Para além do cumprimentos dos prazos de entrega, existem outros fatores que importam aos fabricantes e que devem ser tidos em conta na otimização do planeamento [12]. Fatores como os custos de produção, quantidade de *scrap*, tempo de *setup*, sustentabilidade ambiental e qualidade de produção também deve ser tidos em conta no planeamento, fazendo com que este seja um problema de otimização multi-objetivo. Como nem todos os fabricantes possuem as mesmas necessidades, é comum que estes fatores precisem de ser ajustados para cada caso em concreto.

2.2.3 Trabalhos Relacionados

Ao longo dos anos, o problema de planeamento de produção industrial foi alvo de muitos estudos e trabalhos publicados, pois representa um desafio real e que cada vez mais os fabricantes necessitam de soluções eficientes e consistentes para o resolver.

A nível de literatura são explorados diversas vertentes do problema, mas todas partem dos princípios base do JSSP. Existe um conjunto de produções $J = \{J_1, ..., J_n\}$, em que a produção J_i é composta por um conjunto de k operações $O_i = \{O_{i1}, ..., O_{ik}\}$, que são alocadas a um conjunto de máquinas $M = \{M_1, ..., M_m\}$ [20, 40, 2]. A alocação de operações está sujeita a um conjunto de restrições, que por norma são transversais aos diferentes cenários encontrados:

- Todas as produções a alocar são conhecidas e definidas à priori;
- As operações podem estar sujeitas a restrições de precedência de execução;
- Assim que uma operação é iniciada, esta é executada por completo, sem pausas;
- Cada máquina só consegue executar uma operação de cada vez;
- Uma operação só pode ser alocada a uma máquina capaz de realizar essa operação;

- Todas as máquinas estão disponíveis a partir de um certo tempo definido;
- São conhecidos os tempos de processamento de cada operação em cada máquina.

Estas restrições são a base utilizada na maioria das modelações encontradas para o JSSP ou FJSSP [20, 40, 2]. Depois consoante algumas especificidade do problema e objetivos pode ser necessário ajustar o modelo e acrescentar mais restrições, ou até mesmo excluir algumas destas. Por exemplo, a nível de operações o modelo pode ser simplificado, caso cada produção seja composta apenas por uma operação ou nunca exista precedência de operações nas produções.

Devido à existência deste conjunto de restrições de planeamento, uma abordagem de resolução baseada em Programação com Restrições, é uma das formas mais intuitivas de solucionar este tipo de problema, pois permite modelar as restrições de forma declarativa [11]. Nesta caso as restrições seriam as apresentadas anteriormente, por exemplo restrições de capacidade, prazos de entrega, precedência de operações, entre outras. As variáveis destes tipo de problema, representam as máquinas e operações a alocar e por isso possuem domínios finitos [11].

Em muitos deste casos, o problema é modelado como multi-objetivo, pois o objetivo é otimizar vários fatores do planeamento. O objetivo mais comum a ser minimizado neste tipo de problemas é o tempo total de duração do planeamento [12, 2]. No entanto, existem outros objetivos que podem ser igualmente relevantes de otimizar. A minimização dos custos de produção é um dos grandes objetivos aquando da realização do planeamento. Os custos de produção são compostos por vários fatores, que variam consoante a industria em causa. A minimização do tempo total de duração do planeamento pode ir de encontro à minimização de custos, se assumirmos que os custos são diretamente proporcionais à duração de um produção, mas nem sempre é o caso. Consideremos o exemplo presente na Figura 2.5. A máquina M_1 consegue realizar a operação O_1 para 1000 unidades em 8 horas e M_2 consegue realizar o mesmo processo em 7 horas. No entanto, o consumo energético de M_2 é superior, o que faz que com que o consumo total seja menor em M_1, mesmo com o processo a ter uma duração superior.

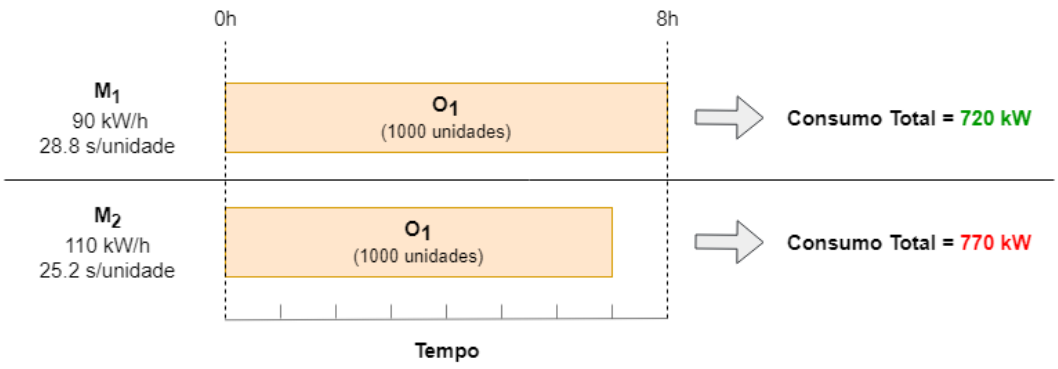

Figura 2.5: Exemplo de comparação entre consumo energético e tempo de processamento.

A minimização do consumo energético é um objetivo bastante comum e encontrado em vários trabalhos publicados. O aumento deste tipo de trabalho também está relacionada com a crescente preocupação ambiental e a procura por soluções que ajudem a tornar os processos de produção mais sustentáveis. Fenómenos como o aquecimento global e o esgotamento de recursos não-renováveis, levaram a que nos últimos anos, o tema de produção sustentável fosse cada vez mais relevante na industria [37, 57].

Nestes casos de minimização do consumo, é utilizada a informação de quanto é o consumo energético de cada máquina ao realizar cada operação, para ser possível calcular o consumo total [40, 20, 57]. Caso não seja possível recolher a informação tão pormenorizada dos processos, o consumo médio da máquina também pode ser utilizado como indicador, principalmente se as operações realizadas forem similares, pois não existem grandes variações de consumo entre operações.

A maioria das modelações realizadas para este tipo de problema não tem em conta pausas de manutenção ou possíveis avarias nas máquinas, mas existem alguns casos onde é tido em conta as manutenções preventivas agendadas. O método mais completo, mas também mais complexo, envolve prever problemas futuros, considerando o histórico de avarias, o tempo que decorre entre manutenções e as horas de trabalho da máquina [40]. Isto leva a que seja possível ajustar o planeamento de produção e agendar manutenções preventivas, e evitar que a produção seja forçada a parar devido a um avaria.

O objetivo da implementação pode ser gerar um planeamento otimizado de raiz ou otimizar um planeamento já existente. Até agora foram focadas mais as soluções que envolvem gerar um planeamento otimizado de raiz, mas a funcionalidade de otimizar um planeamento já existente é bastante útil e muitas vezes aplicável em cenários reais, visto que permite aos fabricantes reajustar um planeamento caso ocorra alguma avaria. Nestes casos, não é pretendido fazer alterações profundas ao planeamento inicial, apenas devem ser realizados alguns ajustes que melhorem o mesmo. Este fator pode ser acrescentado à função objetivo do modelo, otimizando assim a estabilidade do planeamento gerado em relação ao planeamento inicial. A estabilidade de um planeamento é calculada como a média da diferença entre os tempos de produção do planeamento gerado e do inicial [2].

Depois de modelado o problema de planeamento, com todas as especificidades e nível de complexidade adequado ao objetivo que é pretendido alcançar e as necessidades a serem colmatadas, é necessário desenvolver o método que permite para obter uma solução otimizada. Na maioria dos casos recorre-se a meta-heurísticas, baseadas em populações de soluções ou o melhoramento iterativo de uma solução, para resolver o problema de planeamento industrial, pois estes métodos conseguem realizar uma procura vasta pelo espaço de soluções em tempo útil. Para além dos Algoritmos Genéticos, com alguma frequência também são encontrados na literatura os seguintes métodos de otimização, aplicados ao JSSP:

Particle Swarm Otimization (PSO)

O PSO é inspirado no comportamento e movimentação de cardumes de peixes e bandos de pássaros [43]. Este método permite minimizar ou maximizar uma função objetivo de forma iterativa, através de um população de soluções (partículas) [16, 19]. Esta população de partículas é composta por soluções geradas de forma aleatória. A cada partícula é atribuída uma velocidade e a cada iteração as partículas são movimentadas no espaço de soluções de acordo com essa velocidade [43, 16]. A velocidade e trajetória de cada partícula são influenciadas pela melhor solução global encontrada e a melhor posição encontrada atingida pela respetiva partícula [16, 43]. Quando um certo critério de paragem for atingido, como um limite de iterações, é retornada a melhor solução encontrada no processo [19].

Ant Colony Optimization (ACO)

O ACO tenta replicar o comportamento de colónias de formigas na procura por alimento, de modo a resolver um certo problema de otimização. Este algoritmo utiliza uma população de "formigas", em que cada uma representa uma solução, para procurar a melhor solução dentro do espaço de soluções [21, 56]. De forma iterativa, as formigas movimentam-se dentro de espaço de soluções à procura de melhorar as soluções. A movimentação de cada formiga é influenciada pelas "feromonas"deixadas por outras formigas. Quanto melhor uma solução é, mais feromonas são deixadas no local, fazendo com que as formigas sejam atraídas para os arredores das melhores soluções [21]. Este processo é repetido até que seja atingido um certo critério de paragem, como um limite de iterações, e no fim é retornada a melhor solução encontrada [62].

Simulated Annealing (SA)

O SA é inspirado no processo realizado na industria metalúrgica, que consiste no aquecimento a altas temperaturas do metal e arrefecimento lento do mesmo, de modo a obter uma estrutura com o mínimo de fraturas e irregularidades [8]. No início da execução é gerada apenas uma solução base. Em seguida é gerada uma solução vizinha da solução base. A definição de vizinhança depende das características do problema, mas no caso do planeamento de produção, por exemplo, a solução gerada pela ação de trocar duas produções pode ser considerada uma solução vizinha. Se a solução vizinha for melhor que a solução base, então passa a ser a nova solução base. Mesmo que a solução vizinha não seja melhor que a solução base, esta pode tornar-se na nova solução base, dependente da diferença entre os valores da função objetivo e o valor de temperatura. Quanto maior o valor de temperatura maior a probabilidade de aceitar piores soluções como soluções base. A temperatura é definida inicialmente com um valor alto, e este vai diminuindo a cada iteração, de modo a evitar ótimos locais [8, 17]. Este processo é repetido de forma iterativa e quando for atingido um certo critério de paragem é retornada a melhor solução

encontrada.

Tabu Search (TS)

Este é um método de otimização que realiza uma procura no espaço de soluções possíveis, enquanto mantém um histórico das soluções já encontradas. Este histórico de soluções já encontradas é a "lista tabu". Esta lista impede que o algoritmo retorne a soluções já verificadas anteriormente [14, 18]. Inicialmente é gerada uma solução base. Em seguida, é gerado um conjunto de soluções vizinhas da base. Se alguma das soluções vizinhas for melhor que a base, então essa torna-se na nova base e é acrescentada à lista tabu [14]. A lista possui um tamanho limite e sempre que esse limite é atingido. a solução mais antiga é removida da lista para acrescentar uma nova. Este processo de pesquisa é repetido iterativamente, até ser satisfeita alguma condição de paragem. No final é retornada a melhor solução encontrada [14].

Assim, o facto problema de planeamento de produção ser *NP-hard* leva a que seja necessário recorrer a métodos de otimização, como os já apresentados, para obter soluções satisfatórias em tempo útil. A utilização de combinações entre vários destes algoritmos para obter melhores resultados é uma técnica interessante e que cada vez tem sido mais explorada, de modo a obter melhores resultados. Por exemplo, a utilização do PSO para gerar a população inicial de um Algoritmo Genético ou SA/TS para melhorar as soluções. Todas estas diferentes abordagens, tanto o uso isolado ou combinado de algoritmos, podem ser utilizadas para solucionar este tipo de problema. No entanto, a modelação realizada do problema, principalmente a definição da função objetivo, é fundamental na obtenção de bons resultados.

3

Desenvolvimento e Implementação

Neste capítulo é abordada a fase de desenvolvimento do trabalho. Primeiramente, é apresentado o método de recolha de dados utilizado para alimentar o algoritmo. Em seguida é apresentado a evolução incremental do modelo do problema ao longo do trabalho. Por fim, é apresentada a implementação realizada do Algoritmo Genético e as suas diferentes versões desenvolvidas.

3.1 Recolha de Dados

De modo a ser possível realizar o planeamento de produção de uma fábrica, seja por meio de Algoritmos Genéticos ou outro tipo de abordagem, é necessário ter acesso a um conjunto de dados relacionados com o funcionamento e características da fábrica e as suas respetivas produções.

Para tal, é fundamental existir uma recolha de dados, normalmente suportada por um *MES* (*Manufacturing Execution System*) [27], como é o caso do *RAILES*. O objetivo deste tipo de sistemas é digitalizar ao máximo os processos do dia-a-dia de uma fábrica, de forma a aproximar a mesma o máximo possível do conceito de *smart factory*, que consiste na ideia em que toda a fábrica está incorporada num sistema, onde existe a recolha, transmissão e tratamento de dados, em tempo real, acompanhada de um processo de simulação e otimização que visam melhorar o funcionamento da fábrica a diversos níveis [36].

De modo geral, um sistema do tipo *MES* permite interligar a parte operacional do chão de fábrica com a gestão de topo da fábrica [27].

Estes dados recolhidos estão assim relacionados com as diferentes componentes da fábrica, podendo ser dados referentes às máquinas e produções, recolhidos automaticamente por meio de sensores, ou informação de carácter logístico como encomendas, turnos e pausas que são introduzidos no sistema *MES* pelo responsável da fábrica.

Junto com a *Muvu Technologies*, foi realizado um levantamento de requisitos, objetivos a cumprir, necessidades e quais as funcionalidades chave que vão de encontro às reais necessidades dos fabricantes e que podem ser suportadas no *RAILES*. Como resultado desta análise, definiu-se que os dados e informações recolhidos e necessários para o

funcionamento do algoritmo baseiam-se em cinco principais vertentes: Turnos, Pausas, Produções/Encomendas, Produtos e Máquinas.

A nível de Turnos é fundamental ter a informação de quando a fábrica está a funcionar, pois caso contrário é impossível realizar um planeamento de produção. Quanto às Pausas, a razão para ser necessário ter informação das mesmas é idêntica aos Turnos, no entanto existe uma característica que os difere, que é o facto de que enquanto um turno afeta a fábrica toda uma pausa pode afetar apenas uma certa máquina, por exemplo por motivos de uma manutenção agendada.

Quanto às Máquinas é necessário saber a partir de que momento cada máquina está disponível para ser planeada, ou seja, quando termina de produzir o que já tem planeado anteriormente, e quais os gastos energéticos das mesmas, pois o consumo energético é um dos aspetos a ser avaliado na tentativa de minimizar o mesmo, através do planeamento otimizado obtido.

Na vertente dos Produtos é importante para cada produto saber quais máquinas estão aptas a produzir o mesmo e o ciclo de tempo de produção por unidade, tempo de *setup* e *scrap rate* do produto em cada máquina. Outro aspeto fundamental é saber quantas produções em simultâneo podem ocorrer de cada produto. Como estamos a abordar um problema na industria de produção por injeção de plástico, este fator reflete-se no número de moldes disponíveis para cada produto. Por exemplo, se a fábrica tiver dois moldes para um certo produto, então duas máquinas podem estar a produzir o mesmo em simultâneo. A acrescentar a estas informações é relevante saber qual a quantidade mínima a produzir de um certo produto cada vez que se inicia uma produção do mesmo.

Quanto às Produções/Encomendas é fulcral ter acesso ao *deadline* (data de entrega) de cada encomenda, ao produto em causa e quantidade a produzir.

O RAILES disponibiliza uma grande variedade de ferramentas e funcionalidades aos utilizadores, onde estes dados também são necessários, desde a monitorização de processos e qualidade, planeamento logístico, gestão de recursos, etc. Devido a isto, o sistema já possui uma estrutura, tanto de recolha como tratamento e armazenamento deste tipo de dados, o que facilitou a obtenção dos mesmos para executar o algoritmo.

3.2 Formulação do Problema

Quando tratamos problemas de otimização é fundamental a definição do modelo que formula o problema. Um modelo é composto por variáveis, função objetivo e restrições.

Um mesmo caso real pode ser modelado de diferentes formas, o que leva a que matematicamente o problema seja diferente, apesar de na prática tratar-se do mesmo problema real. Posto isto, um dos principais desafios foi definir o modelo a utilizar que representasse da melhor forma o tipo de problema de planeamento industrial já apresentado.

Ao longo do desenvolvimento do projeto, o modelo inicialmente proposto sofreu alterações. A principal e mais profunda alteração ocorreu devido a problemas de performance

na execução do algoritmo. Depois de implementado e feitos alguns testes do algoritmo com o modelo inicial, verificou-se que o tempo de execução do mesmo era demasiado elevado e que a forma como tinha sido definido o modelo seria o principal responsável e causador desse problema na performance do algoritmo.

Este fator levou a que fosse necessário repensar o modelo, de forma a melhorar a performance do algoritmo. Depois de alterado o modelo, foram acrescentadas mais restrições e ajustada a função objetivo de modo a torná-lo mais completo e melhor representar o problema real de planeamento, conseguindo assim encontrar soluções que sejam úteis e aplicáveis em casos industriais reais.

Nesta secção é primeiro apresentado o modelo inicial utilizado e em seguida o modelo revisto e as suas principais diferenças, juntamente com as alterações incrementais que foram realizadas ao longo da realização do projeto.

3.2.1 Modelo Inicial

Ainda durante o período de preparação, foi desenvolvida uma modelação inicial para o problema de planeamento de produção industrial já descrito. Esta modelação não refletia o problema na sua totalidade, pois ainda não incorporava algumas vertentes, como é o casos dos prazos de entrega ou a quantidade de moldes por produto, no caso da produção por injeção de plástico. No entanto, esta modelação foi um ponto de partida para solucionar o problema e chegou a ser implementada numa primeira fase.

Foram identificados três fatores determinantes na representação de um planeamento industrial: as máquinas, os produtos/operações que serão alocados às mesmas e as ordem com que são executadas. Para um problema com um conjunto de máquinas $i = 0, ..., I$, produtos $j = 0, ..., J$ e ordens de produção $k = 0, ..., K$, o modelo é composto pelas variáveis x_{ijk}, e_{ijk} e q_{ijk}. A matriz x_{ijk} e e_{ijk} são matrizes de valores binários (1 ou 0), que representam as alocações de produções pelas diferentes ordens de produção das máquinas e as pausas de *Setup* nas produções, respetivamente. A matriz q_{ijk} representa as quantidades a produzir em cada produção alocada, e por isso, possui valores inteiros superiores ou iguais a zero devido à natureza do problema, onde não é possível produzir quantidades negativas de produtos, meias unidades e não existe precedência de componentes, ou seja, unidades que são compostas por diferentes componentes que são produzidas separadamente e por determinada ordem.

Nesta abordagem o objetivo é minimizar o consumo energético das máquinas utilizadas na produção. Para tal, a função objetivo, é representada pela soma do consumo de todas as máquinas durante a produção. Este consumo é obtido através da multiplicação do consumo energético médio da máquina C_i pelo respetivo tempo de produção D_i.

$$Min\ C = \sum_{i=1}^{I} (C_i \times D_i) \qquad (1)$$

$$D_i = \sum_{j=1}^{J} \sum_{k=1}^{K} (x_{ijk} \times (q_{ijk} \times T_{ij} + P_{ij} \times e_{ijk})) \qquad (2)$$

É importante garantir que as quantidades de produtos encomendados são produzidas, de modo a cumprir as encomendas. Ao realizar esta verificação deve-se ter em conta os valores de *Scrap Rate* da cada máquina com cada produto S_{ij}, pois as unidades com defeito são descartadas e não podem ser contabilizadas a nível de encomendas.

$$\sum_{i=1}^{I} \sum_{k=1}^{K} \frac{q_{ijk}}{(1 + S_{ij})} \geq Q_j \ , \quad j = 1, ..., J \qquad (3)$$

Também é necessário garantir que numa certa posição da sequência de produção de uma máquina, apenas pode estar a ser produzido um tipo de produto e que a respetiva máquina consegue produzir esse produto.

$$\sum_{j=1}^{J} x_{ijk} \leq 1 \times E_{ij} \ , \quad i = 1, ..., I \ , \ k = 1, ..., K \qquad (4)$$

De forma similar, é necessário garantir que em cada posição na sequência de produção de uma máquina, só existe no máximo uma pausa de *Setup* antes de começar a produção de um certo produto.

$$\sum_{j=1}^{J} e_{ijk} \leq 1 \ , \quad i = 1, ..., I \ , \ k = 1, ..., K \qquad (5)$$

Tendo em conta as variáveis, função objetivo e restrições apresentadas, o modelo inicialmente definido é representado na íntegra da seguinte forma:

Função Objetivo :

$$Min\ C = \sum_{i=1}^{I} (C_i \times D_i) \qquad (1)$$

$$D_i = \sum_{j=1}^{J} \sum_{k=1}^{K} (x_{ijk} \times (q_{ijk} \times T_{ij} + P_{ij} \times e_{ijk})) \qquad (2)$$

s.a :

$$\sum_{i=1}^{I} \sum_{k=1}^{K} \frac{q_{ijk}}{(1 + S_{ij})} \geq Q_j \ , \quad j = 1, ..., J \qquad (3)$$

$$\sum_{j=1}^{J} x_{ijk} \leq 1 \times E_{ij} \ , \quad i = 1, ..., I \ , \ k = 1, ..., K \qquad (4)$$

$$\sum_{j=1}^{J} e_{ijk} \leq 1 \ , \quad i = 1, ..., I \ , \ k = 1, ..., K \qquad (5)$$

$$x_{ijk}\ , e_{ijk} \in \{0,1\} \ \wedge \ q_{ijk} \in \mathbb{Z}_0^+ \qquad (6)$$

Onde :

i = índice de Máquina
j = índice de Produto
k = índice na sequência de produção
I = número total de máquinas
J = número total de produtos
K = número total de ordens de produção
S_{ij} = *scrap rate* da máquina i para o produto j
Q_j = quantidade total de produto j a produzir
E_{ij} = se a máquina i é capaz de produzir o produto j (valor de 0 ou 1)
C_i = consumo energético médio da máquina i
D_i = tempo de produção da máquina i
T_{ij} = tempo de ciclo para produzir uma unidade do produto j na máquina i
P_{ij} = tempo de pausa ou preparação para iniciar a produção de j na máquina i
x_{ijk} = produzir o produto j na máquina i na posição k da sequência de produção
q_{ijk} = quantidade de produto j produzido na máquina i na posição k da sequência de produção
e_{ijk} = existência de pausa antes de começar a produção de j na máquina i na posição k da produção

Esta modelação inicial foi desenvolvida para dar resposta as necessidades mais básicas do problema de planeamento apresentado e este foi o ponto de partida para o desenvolvimento.

Como já foi referido, foi identificado um problema de performance relacionado com este modelo, ou seja, o tempo de execução do algoritmo era demasiado elevado e não era possível obter bons resultados em tempo útil.

Previsivelmente, o tempo de execução escala consoante a dimensão do problema, que é traduzido no número de máquinas, produtos e ordem de produção. Estes 3 fatores são os que mais influenciam o tempo de execução com este modelo, pois são os índices das variáveis x_{ijk}, e_{ijk} e q_{ijk}. Quanto maiores os valores destes índices, maior o número de iterações em cada ciclo e operações, o que traduz-se num maior tempo de execução.

Na Tabela 3.1 são apresentados os tempos de execução médios do algoritmo com este modelo inicial, para diferentes casos com uma população de 100 indivíduos e 200 gerações de execução. Os testes realizados foram executados numa máquina com um processador Intel i7-8750H (2.20GHz) e 16GB de memória RAM.

Máquinas (i)	**Produtos (j)**	**Ordens de Produção (k)**	**Tempo por Geração**
3	5	5	0.165s
10	20	15	2.75s
30	50	20	12.65s

Tabela 3.1: Tempos de execução do algoritmo com o modelo inicial

Ao analisar estes valores, podemos notar que para exemplos de pequenas dimensões o tempo de execução é razoável, no entanto, conforme aumenta a complexidade do problema o tempo de execução também aumenta consideravelmente. Com base nestes resultados, considerou-se que não seria viável continuar a implementação da solução com este modelo do problema, tendo em conta que este modelo inicial não contempla todas as restrições e especificidades do problema de planeamento de produção (verificação de datas de entrega, restrição de produções em simultâneo, etc.), e que estas acrescentam complexidade ao modelo, que se traduzem num maior tempo de computação.

Posto isto, foi necessário rever e reestruturar o modelo de forma a colmatar este problema do elevado tempo de execução e por isso foi desenvolvido o novo modelo.

3.2.2 Modelo Revisto

Este novo modelo difere em alguns pontos do modelo inicial já apresentado, sendo que as alterações foram realizadas com o objetivo de obter melhor performance na execução do algoritmo e acrescentar as restrições e fatores, que não eram contemplados no modelo inicial, mas são necessários para que a solução satisfaça as necessidades e objetivos definidos inicialmente.

O principal objetivo ao realizar a alteração no modelo foi reduzir o número de iterações em cada ciclo, ao realizar as diferentes operações do algoritmo. Para tal, desenvolveu-se uma solução em que foi reduzida a quantidade de índices utilizados nas variáveis x e q, passando assim, a serem utilizados apenas como índices os valores da máquina e da ordem de produção. A variável x_{ik}, continua a definir as alocações de produções, mas passou a uma matriz com valores que correspondem ao índice do produto, em vez de valores binários.

Por exemplo, consideremos a máquina 1, com 3 produtos disponíveis e 4 ordens de produção possíveis, onde vão ser produzidos os produtos 3, 1 e 2, por esta mesma ordem. Em seguida são apresentadas duas matrizes, a primeira matriz representa o planeamento da máquina 1 no primeiro modelo e a segunda no modelo revisto.

$$x_{1jk} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$x_{1k} = \begin{bmatrix} 3 & 1 & 2 & 0 \end{bmatrix}$$

Com esta alteração, o tamanho da matriz reduz significativamente e também deixa de ser necessário verificar se na mesma máquina estão a ser produzidos dois ou mais produtos em simultâneo. Isto reduz o número de iterações e operações, o que leva a uma redução no tempo de execução.

A variável x_{ik} assume os valores de j ou 0 e q_{ik} possui valores inteiros superiores ou iguais a zero devido à natureza do problema, onde não é possível produzir quantidades negativas de produtos, meias unidades e não existe precedência de componentes,

ou seja, unidades que são compostas por diferentes componentes que são produzidas separadamente e por determinada ordem.

As variáveis t_{ik0} e t_{ik1} assumem valores reais positivos, pois correspondem a datas e horas traduzidas em *timestamps*, e são calculadas tendo em conta x_{ik} e q_{ik}. O cálculo das matrizes t_{ik0} e t_{ik1} é realizado através de um método, que aloca as produções no tempo, tendo em conta turnos, pausas de produção e eventuais conflitos com pausas de produção e com o limite de produções do mesmo produto em simultâneo. Na Figura 3.1 é apresentado o diagrama que explica a lógica que compõem este método.

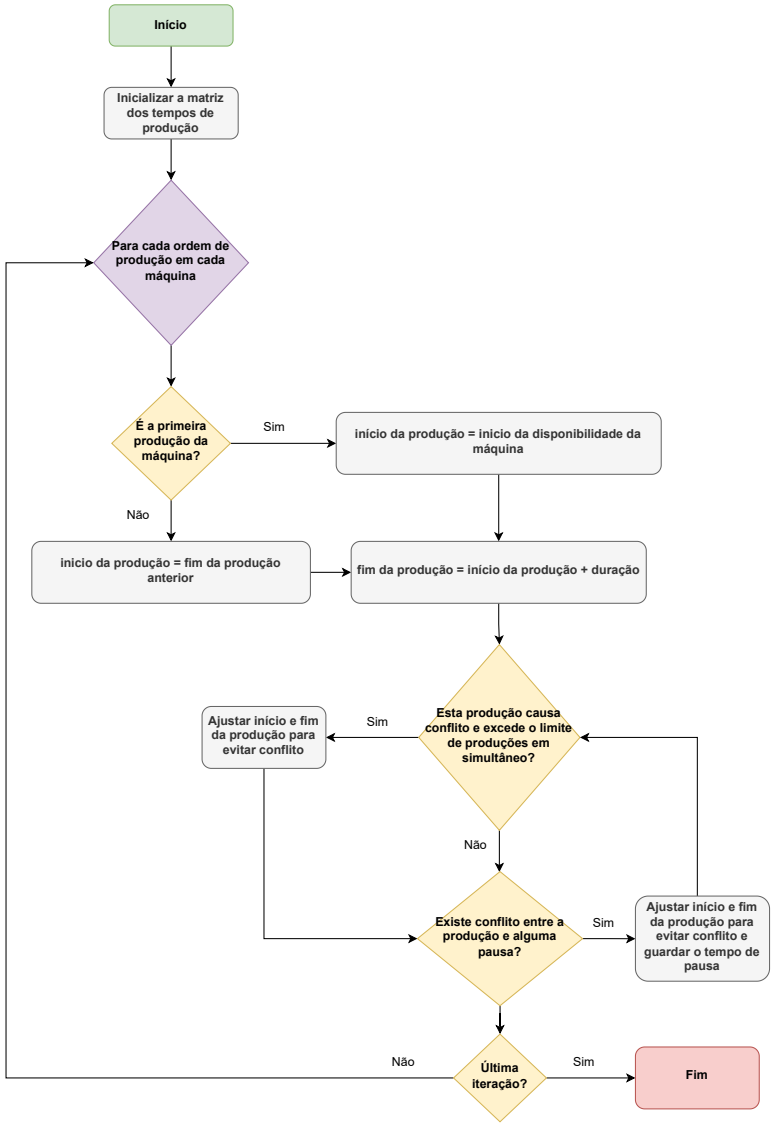

Figura 3.1: Diagrama sobre o cálculo dos tempos de inicio e fim das produções.

As pausas da fábrica não foram representadas no modelo, pois não interferem diretamente na função objetivo ou nas restrições impostas. No entanto, as pausas restringem a

alocação de produções no tempo, ou seja, são um dos fatores que interferem no cálculo das matrizes t_{ik0} e t_{ik1}. A resolução de conflitos entre produções e pausas é fundamental para tornar o planeamento exequível. Na Figura 3.2 são ilustrados alguns casos de conflito e a respetiva resolução dos mesmos.

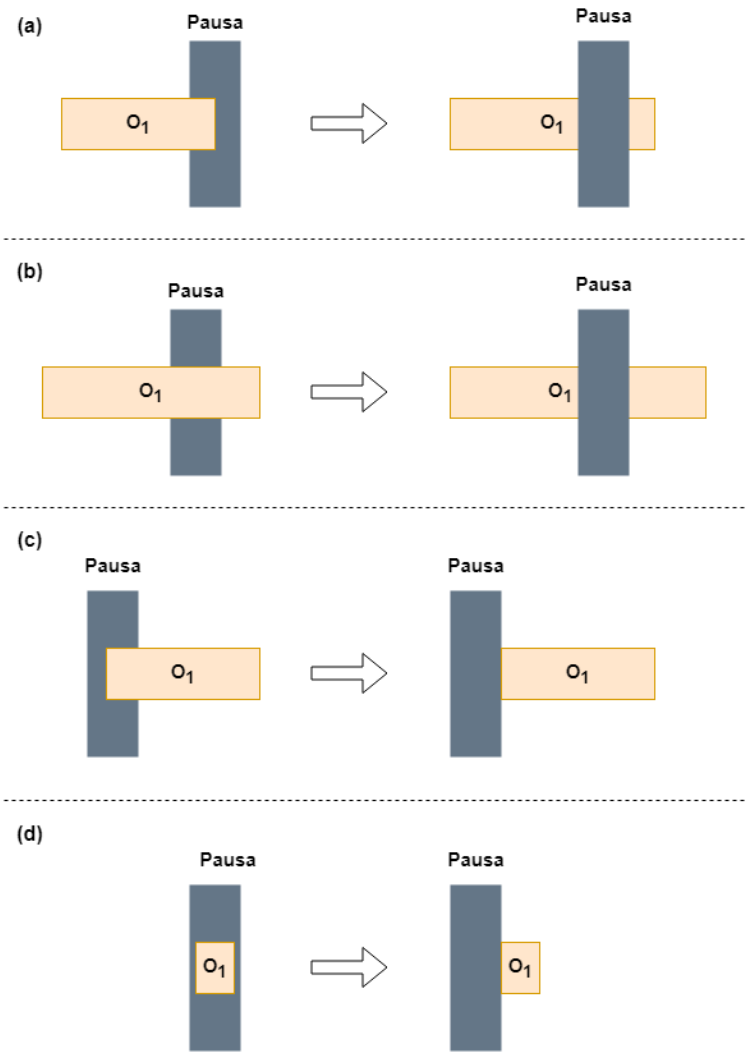

Figura 3.2: Exemplo da resolução de conflitos entre operações e pausas.

A verificação de conflitos entre produções e pausas é feita por ordem crescente cronologicamente e os conflitos são sempre resolvidos de forma ascendente, ou seja, avançando as produções na linha temporal. Para cada produções e pausa é realizada a verificação da existência de conflito e resolução do mesmo, se necessário. Representando o início e fim da produção como t_1 e t_2 e o início e fim da pausa como p_1 e p_2, ambos respetivamente, o método utilizado pode ser representado da seguinte forma:

$$f(t_1, t_2, p_1, p_2) = \begin{cases} (t_1, p_2 + t_2 - p_1), & \text{se } t_1 < p_1 \wedge t_2 > p_1 \\ (p_2, p_2 + t_2 - t_1), & \text{se } t_1 < p_2 \wedge t_1 > p_1 \\ (t_1, t_2), & \text{se caso contrário} \end{cases}$$

Para além desta alteração a nível de variáveis do modelo, ao longo do desenvolvimento do projeto, também foram realizadas alterações de modo a completar o modelo e aproximar o mesmo o mais possível da realidade.

Neste novo modelos o objetivo para além de minimizar o consumo energético das máquinas utilizadas na produção, também é minimizar a quantidade de *scrap*, o tempo de *setup* e cumprir as datas de entrega das encomendas. Para tal, a função objetivo, apresentada na equação 1, pode ser dividida nestas quatro componentes distintas. A cada uma destas quatro componentes pode ser atribuído um peso diferente. Estes pesos são representados por W_e, W_{scrap}, W_{setup} e W_d, que correspondem ao peso do consumo, *scrap*, *setup* e cumprimento de *deadline*, respetivamente. O facto de os pesos serem ajustáveis tem o objetivo de tornar o modelo o mais adaptável possível a diferentes cenários, pois, por exemplo, uma certa fábrica pode colocar mais importância ao consumo energético mas uma outra pode dar mais importância à quantidade de *scrap*.

$$\text{Min } F = Energy \cdot W_e + Scrap \cdot W_{scrap} + Setup \cdot W_{setup} + W_d \cdot Deadline \quad (1)$$

O consumo energético total é obtido através da soma do consumo de cada máquina, em que este é calculado através da multiplicação do consumo energético médio da máquina C_i pelo respetivo tempo de produção, representado na equação 2.

$$Energy = \sum_{i=1}^{I} \sum_{k=1}^{K} \left(C_i \cdot q_{ik} \cdot T_{ix_{ik}} \right) \quad (2)$$

A número de unidades de *scrap* de uma máquina é obtido através da quantidade produzida e do *scrap rate* de cada produto nessa máquina, representado na equação 3.

$$Scrap = \sum_{i=1}^{I} \sum_{k=1}^{K} \left(q_{ik} - \frac{q_{ik}}{(1 + S_{ix_{ik}})} \right) \quad (3)$$

O tempo total de *setup* de uma máquina é a soma dos tempos de *setup* das produções efetuadas nessa máquina, que corresponde à equação 4.

$$Setup = \sum_{i=1}^{I} \sum_{k=1}^{K} P_{ix_{ik}} \quad (4)$$

Em relação ao cumprimento das datas de entrega das encomendas este foi incorporado na função objetivo e não como uma restrição, pois pode acontecer não ser logisticamente possível cumprir todas as datas de entrega, mas mesmo assim, deve ser possível que o

algoritmo produza um planeamento em que tenta minimizar esses atrasos nas encomendas. Para tal, qualquer atraso numa encomenda penaliza o valor da função objetivo. A penalização, representada pela equação 5, é proporcional ao atraso, ou seja, quanto maior o atraso na data de entrega maior a penalização, pois é feita a diferença entre a data real de entrega e o *deadline* da encomenda.

$$Deadline = \sum_{e=1}^{E} \max\left(A_e - \text{Deadline}_e, 0\right) \qquad (5)$$

Como no modelo inicial, é importante garantir que as quantidades de produtos encomendados são produzidas, de modo a cumprir as encomendas, tendo em conta a taxa de *Scrap Rate* da cada máquina com cada produto S_{ij}. Na inequação 6 está representada a restrição referente às quantidades mínimas a serem produzidas.

$$\sum_{i=1}^{I} \sum_{k=1}^{K} \frac{q_{ik}}{(1 + S_{ix_{ik}})} \geq Q_j, \quad j = 1, \ldots, J \qquad (6)$$

A nível da matriz x_{ik}, é importante garantir que uma máquina não tem produções com o mesmo produto seguidas e que o valor 0 assinala o termino da sequência de produção dessa máquina, da forma representada na condição 7. Com esta alteração foi removida a variável que correspondia à existência de pausas, pois assim existem sempre pausas, visto que há sempre troca de produto na sequência de produção. Também é importante restringir a alocação de produções, garantindo, que apenas são alocados produtos cuja máquina está apta a produzir, através da restrição 8.

$$(x_{ik} \neq x_{(ik+1)}) \vee (x_{ik} = 0 \rightarrow x_{(ik+1)} = 0), \quad i = 1, \ldots, I, \quad k = 1, \ldots, K-1 \qquad (7)$$

$$x_{ik} \neq 0 \rightarrow E_{ix_{ik}} = 1, \quad i = 1, \ldots, I, \quad k = 1, \ldots, K-1 \qquad (8)$$

É necessário verificar a quantidade de produções em simultâneo, de um mesmo produto, de modo a que o planeamento seja exequível na realidade. Por exemplo, se existirem três moldes de um produto, então só é possível produzir o mesmo em até três máquinas em simultâneo. A restrição 9 verifica se o número de produções em simultâneo, de um mesmo produto, não ultrapassa o limite imposto.

$$\sum_{i_2=1}^{I} \sum_{k_2=1}^{K} \left\{ \begin{array}{ll} 1, & \text{se } i_1 \neq i_2 \wedge x_{i_1k_1} = x_{i_2k_2} \wedge t_{i_1k_10} < t_{i_2k_21} \wedge t_{i_1k_11} > t_{i_2k_20} \\ 0, & \text{se caso contrário} \end{array} \right\} \leq N_{x_{i_1k_1}} - 1,$$

$$i_1 = 1, \ldots, I, \quad k_1 = 1, \ldots, K \qquad (9)$$

Em seguida é apresentado, por completo, o modelo revisto já descrito.

Função Objetivo :

$$\text{Min } F = Energy \cdot W_e + Scrap \cdot W_{scrap} + Setup \cdot W_{setup} + W_d \cdot Deadline \quad (1)$$

$$Energy = \sum_{i=1}^{I} \sum_{k=1}^{K} \left(C_i \cdot q_{ik} \cdot T_{ix_{ik}} \right) \quad (2)$$

$$Scrap = \sum_{i=1}^{I} \sum_{k=1}^{K} \left(q_{ik} - \frac{q_{ik}}{(1 + S_{ix_{ik}})} \right) \quad (3)$$

$$Setup = \sum_{i=1}^{I} \sum_{k=1}^{K} P_{ix_{ik}} \quad (4)$$

$$Deadline = \sum_{e=1}^{E} \max\left(A_e - \text{Deadline}_e, 0\right) \quad (5)$$

s.a:

$$\sum_{i=1}^{I} \sum_{k=1}^{K} \frac{q_{ik}}{(1 + S_{ix_{ik}})} \geq Q_j, \quad j = 1, \ldots, J \quad (6)$$

$$(x_{ik} \neq x_{(ik+1)}) \vee (x_{ik} = 0 \rightarrow x_{(ik+1)} = 0), \quad i = 1, \ldots, I, \quad k = 1, \ldots, K-1 \quad (7)$$

$$x_{ik} \neq 0 \rightarrow E_{ix_{ik}} = 1, \quad i = 1, \ldots, I, \quad k = 1, \ldots, K \quad (8)$$

$$\sum_{i_2=1}^{I} \sum_{k_2=1}^{K} \left\{ \begin{array}{ll} 1, & \text{se } i_1 \neq i_2 \wedge x_{i_1k_1} = x_{i_2k_2} \wedge t_{i_1k_10} < t_{i_2k_21} \wedge t_{i_1k_11} > t_{i_2k_20} \\ 0, & \text{se caso contrário} \end{array} \right\} \leq N_{x_{i_1k_1}} - 1,$$

$$i_1 = 1, \ldots, I, \quad k_1 = 1, \ldots, K \quad (9)$$

$$x_{ik} \in \{0\} \cup j, \quad q_{ik} \in \mathbb{Z}_0^+, \quad t_{ik0}, t_{ik1} \in \mathbb{R}_0^+ \quad (10)$$

i = índice de Máquina
j = índice de Produto
k = índice na sequência de produção
e = índice de encomenda
I = número total de máquinas
J = número total de produtos
K = número total de ordens de produção
E = número total de encomendas
W_e = peso do consumo energético na função objetivo
W_{setup} = peso do tempo de setup na função objetivo
W_{scrap} = peso da quantidade de scrap na função objetivo
W_d = peso do cumprimento da data de entrega das encomendas na função objetivo
a_e = data em que a encomenda e estará pronta
$Deadline_e$ = data de entrega da encomenda e
N_j = número máximo de máquinas que conseguem produzir o produto j em simultâneo
S_{ij} = *scrap rate* da máquina i para o produto j
Q_j = quantidade total de produto j a produzir
E_{ij} = se a máquina i é capaz de produzir o produto j (valor de 0 ou 1)
C_i = consumo energético médio da máquina i
$Scrap$ = quantidade de *scrap* total
$Setup$ = o tempo de *setup* total
$Energy$ = o consumo energético total
$Deadline$ = soma dos atrasos nas encomendas
T_{ij} = tempo de ciclo para produzir uma unidade do produto j na máquina i
P_{ij} = tempo de pausa ou preparação para iniciar a produção de j na máquina i
x_{ik} = produto j a produzir na máquina i na posição k da sequência de produção
q_{ik} = quantidade a produzir na máquina i na posição k da sequência de produção
t_{ik0} = tempo de inicio da produção na máquina i na posição k da sequência de produção
t_{ik1} = tempo de fim da produção na máquina i na posição k da sequência de produção

Este modelo foi o escolhido para ser implementado em definitivo, em detrimento do modelo inicial. As alterações realizadas permitiram obter uma melhoria a nível de performance e tornar o modelo mais completo, representando melhor a realidade do problema de planeamento de produção e colmatando as necessidades dos fabricantes.

3.3 Algoritmo Genético

Foi realizada a implementação de um Algoritmo Genético de modo a resolver o problema de planeamento de produção industrial.

Quanto à execução do algoritmo, esta segue o padrão clássico dos Algoritmos Genéticos. Em primeiro lugar é gerada a população inicial e depois o processo de elite, seleção,

crossover e mutação são repetidos até ser atingido um certo critério de paragem.

Na Figura 3.3 é apresentado o diagrama com a sequência de processos realizados na execução do algoritmo.

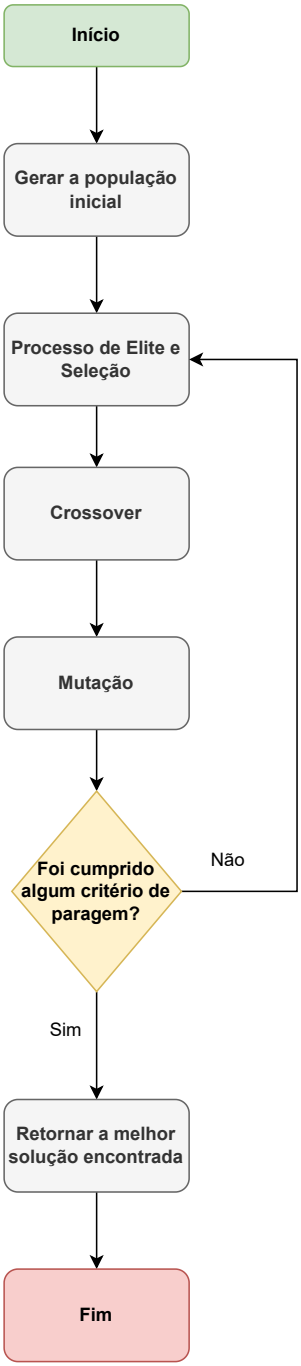

Figura 3.3: Diagrama com a sequência de processos do algoritmo.

Neste capitulo são apresentados os diferentes processos e componentes que compõem o Algoritmo Genético. Para cada processo é apresentada a sua metodologia e principais decisões de implementação tomadas. Por último, são propostas algumas versões alternativas do algoritmo, com o objetivo de serem testadas e comparadas, de modo a determinar

qual a que melhor se adequa para o problema em questão.

3.3.1 Cromossoma

No contexto de algoritmos genéticos, um cromossoma representa uma solução do problema. Cada cromossoma é composto por um conjunto de genes, em que cada um destes corresponde a uma das variáveis do problema.

A base do cromossoma são as matrizes x_{ik} e q_{ik}, que definem a alocação de produções e quantidade, respetivamente. A partir destas é calculada a matriz t_{ik}, que define os tempos de inicio e fim de cada produção, pelo método descrito na Figura 3.1.

Para facilitar o processo de *crossover* entre soluções, é utilizado um formato *string* do cromossoma. Neste formato de *string* cada gene é separado por "|"e apenas são representados os genes das variáveis x_{ik} e q_{ik}, visto que a variável t_{ik} é calculada a partir destas. Em seguida é apresentado um exemplo de matrizes x_{ik} e q_{ik} e o respetivo cromossoma.

$$x_{ik} = \begin{bmatrix} 3 & 12 & 7 & 0 \\ 8 & 2 & 0 & 0 \end{bmatrix}$$

$$q_{ik} = \begin{bmatrix} 200 & 50 & 300 & 0 \\ 350 & 400 & 100 & 0 \end{bmatrix}$$

$$Cromossoma = \text{"03|12|07|00|08|02|00|00|200|050|300|000|350|400|100|000"}$$

3.3.2 Metodologia e Processos

Um algoritmo genético é composto por quatro principais fases na sua execução. Cada uma destas representa um papel importante na execução do algoritmo e impactam diretamente os resultados obtidos pelo mesmo. Nesta secção são apresentadas as principais decisões de implementação nas diversas etapas que compõem o algoritmo genético.

3.3.2.1 Inicialização

Para inicializar a execução do algoritmo, é necessário gerar uma população inicial de soluções. O processo de gerar uma solução é em grande parte aleatório, mas dentro desta aleatoriedade são controlados alguns fatores, de forma a garantir que as soluções geradas cumprem as restrições do problema. Não existindo este controlo, a grande maioria dos indivíduos gerados não seriam soluções do problema e seriam descartados, pois não cumpririam as restrições do mesmo. Isto levaria a um aumento no tempo necessário para gerar a população e tornaria este processo pouco consistente, pois para certos exemplos podia ser rápido e para outros ser necessário mais tempo de execução, sendo que no limite poderia não ser possível gerar a população em tempo útil, pois seria um processo totalmente aleatório.

Para obter uma solução é necessário gerar as matrizes x_{ik} e q_{ik}. Primeiro é necessário gerar a matriz x_{ik}, que corresponde à alocação de produções, e depois são gerados os valores das respetivas quantidades na matriz q_{ik}.

Ao alocar as produções, é importante tentar manter um equilíbrio a nível de que com frequência um certo produto é produzido. Esta preocupação também resulta do facto de as fábricas muitas vezes terem valores mínimos de unidades a produzir por produto, pois pode não fazer sentido iniciar uma produção de um certo produto se o número de unidades não o justificar. Por exemplo, se existe uma encomenda de 10000 unidades de um certo produto e a quantidade mínima de produção do mesmo são 1000 unidades, então logo à partida já sabemos que a solução pode ter até 10 produções alocadas daquele produto. Tendo em conta este fator, a alocação de produções é baseada num sistema de probabilidades, que vão sendo calculadas e atualizadas à medida que a matriz é preenchida, tendo por base as quantidades das encomendas, quantidades mínimas de produção, número produções de cada produto já alocadas e entradas restantes da matriz que faltam preencher (por exemplo: pode faltar alocar produções em 3 máquinas, mas um certo produto só pode ser produzido numa delas, então para esse produto as entradas restantes correspondem apenas às dessa mesma máquina). Este método permite manter a aleatoriedade do processo, mas de forma controlada.

Considerando C_j como o número de produções do produto j já alocadas na matriz e L_j o número de entradas restantes da matriz, em que j pode ser alocado, a probabilidade do produto j ser alocado à máquina i na ordem de produção k pode ser calculada pela seguinte fórmula:

$$
\text{p(i,j,k)} = \begin{cases}
0, & \text{se } E_{ij} = 0 \vee C_j = Q_j/(MinProd_j) \vee (k > 1 \wedge (x_{ik-1} = 0 \vee x_{ik-1} = j)) \\
1/\sum_{j_1=1}^{J} \begin{cases} 1, & \text{se } E_{ij_1} = 1 \wedge C_{j_1} = 0 \wedge L_{j_1} \leq K \\ 0, & \text{se caso contrário} \end{cases}, & \text{se } E_{ij} = 1 \wedge C_j = 0 \wedge L_j \leq K \\
1/\left(\sum_{j_1=1}^{J}(E_{ij_1}) - \sum_{j_1=1}^{J} \begin{cases} 1, & \text{se } E_{ij_1} = 1 \wedge C_{j_1} = Q_{j_1}/(MinProd_{j_1}) \\ 0, & \text{se caso contrário} \end{cases} \right), & \text{se } k > 1 \\
1/\left(\sum_{j_1=1}^{J}(E_{ij_1}) + 1 - \sum_{j_1=1}^{J} \begin{cases} 1, & \text{se } E_{ij_1} = 1 \wedge C_{j_1} = Q_{j_1}/(MinProd_{j_1}) \\ 0, & \text{se caso contrário} \end{cases} \right), & \text{se } k = 1
\end{cases}
$$

A probabilidade de não ser alocado nenhum produto, e terminar assim a sequência de produção, é equivalente ao resto da soma das restantes probabilidades calculadas.

$$p(i, 0, k) = 1 - \sum_{j=1}^{J} p(i, j, k)$$

Com base nas probabilidades calculadas, são alocadas as produções de forma aleatória. Na Figura 3.4 é apresentado o diagrama com o processo de gerar a matriz x_{ik}.

Depois de gerada a matriz x_{ik} é possível gerar a matriz q_{ik} referente às quantidades de produção. Como as quantidades produzidas necessitam de satisfazer as encomendas, então a soma dos valores gerados para cada produto deve ser igual ao necessário para cada produto, já tendo em conta as unidades previstas de *scrap*. Para tal, essa quantidade total de

cada produto é repartida pelas diversas produções do respetivo produto. Esta repartição da quantidade é feita de forma aleatória, tendo em conta os valores das quantidades mínimas de produção e também outro fator que é o incremento de quantidade na produção. Este fator permite que seja definido para cada produto qual o valor de quantidade a incrementar nas produções, por exemplo para um produto com uma quantidade mínima de produção de 1000 unidades e um incremento de 500 unidades, as produções do mesmo poderão ter quantidades de 1000, 1500, 2000, etc. Esta característica foi acrescentada ao algoritmo para que seja possível ter um maior controlo sobre os valores das quantidades nas soluções geradas e que estas mais facilmente vão de encontro às necessidades do problema.

A partir das matrizes x_{ik} e q_{ik} é gerada a matriz dos tempos t_{ik}, utilizando o método já apresentado na Figura 3.1.

Assim, através da repetição desta sequência de processos é possível gerar uma população de soluções inicial, que será o ponto de partida para a execução do algoritmo.

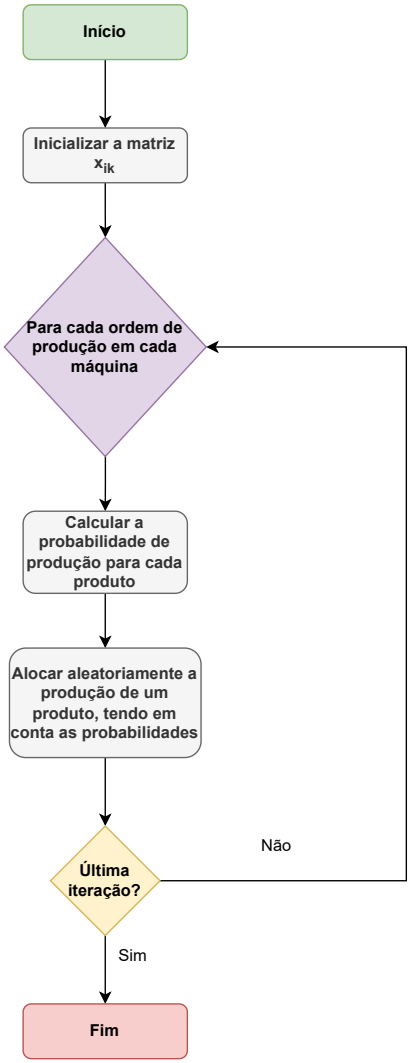

Figura 3.4: Diagrama sobre o processo de gerar a matriz x_{ik}.

3.3.2.2 Seleção e Elite

O processo de seleção determina quais indivíduos da população serão utilizados na fase de reprodução que dará origem à próxima geração. O número de indivíduos selecionados a cada geração é determinado por um valor de proporção para o tamanho da população, previamente definido aquando da inicialização do algoritmo. Foram implementados e testados dois métodos de seleção: método da roleta e o método do torneio.

O método do torneio consiste em escolher aleatoriamente dois indivíduos da população e selecionar o que tem melhor valor de *fitness* dos dois. Este processo é repetido até ser selecionado o número de indivíduos pretendido. Para evitar que possa ocorrer a repetição de indivíduos selecionados, cada indivíduo selecionado é removido do conjunto de selecionáveis, deixando assim de poder ser escolhido. Na Figura 3.5 é apresentado o diagrama do método de seleção de torneio implementado. Na implementação do método da roleta foi necessário fazer algumas adaptações. Como trata-se de um problema de minimização, então é necessário utilizar um valor ajustado de *fitness* para calcular a probabilidade de cada individuo. Este valor ajustado é a diferença entre o valor de *fitness* máximo da população e o valor de *fitness* do indivíduo. Depois é obtida a soma total destes valores e a probabilidade de cada individuo é igual ao valor de *fitness* ajustado dividido pela soma. Na Figura 3.6 é apresentado o diagrama do método de roleta implementado.

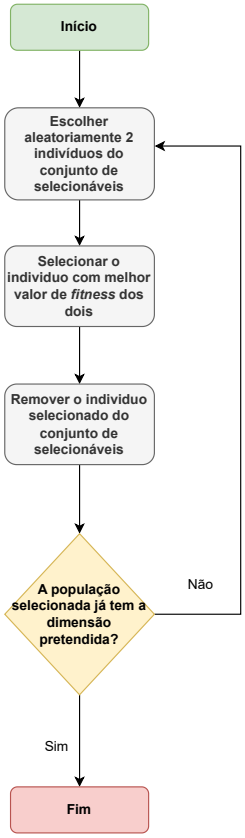

Figura 3.5: Diagrama do método de seleção de torneio.

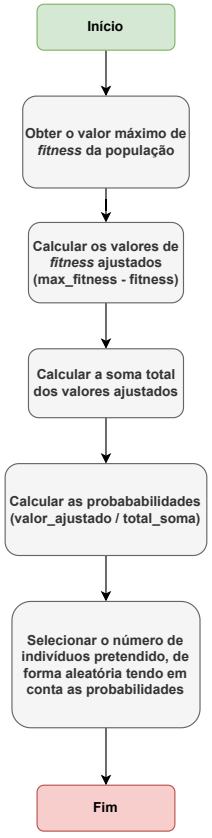

Figura 3.6: Diagrama do método de seleção de roleta.

Também foi implementado o conceito de elite no algoritmo. A elite é um conjunto, composto por uma porção de indivíduos da população com melhor valor de *fitness*. Esta porção é um valor estático definido no início da execução do algoritmo. Este conjunto de indivíduos são automaticamente selecionados sem necessitarem de passar pelo processo de seleção convencional, evitando que as melhores soluções não sejam selecionadas. Assim, o conjunto de indivíduos utilizados na fase de reprodução é composto pela união dos indivíduos da elite com os selecionados por um dos métodos de seleção.

3.3.2.3 Crossover e Mutação

Depois de selecionados os indivíduos da população, estes passam por um processo denominado *crossover*, que consiste na troca de porções de genes entre si, de modo a gerar novos indivíduos.

Um dos desafios neste problema foi definir que genes deviam ser elegíveis para serem trocados entre cromossomas. Analisando a natureza do problema e a modelação efetuada do mesmo, concluiu-se que apenas fazia sentido realizar a operação nos genes pertencentes à matriz de alocação de produções x_{ik}. Isto deve-se ao facto de a matriz das quantidades q_{ik} ser sempre gerada através da matriz x_{ik}, de modo a garantir que a restrição das quantidades mínimas é cumprida. Caso as quantidades também fossem alvo de troca na operação de *crossover*, a maioria dos indivíduos resultantes não seriam considerados soluções, pois não cumpririam a restrição das quantidades mínimas para satisfazer as encomendas. Este fenómeno ocorre pois a matriz q_{ik} é gerada de forma a cumprir essa restrição, e qualquer alteração não calculada inviabilizará a "solução".

Para realizar esta operação são escolhidos aleatoriamente dois indivíduos do conjunto de selecionados na fase de seleção. Depois são definidos de forma aleatória os pontos de corte. São escolhidos dois pontos na estrutura do cromossoma pois foi implementado um método de *crossover* de dois pontos de corte. Como apenas pretendemos realizar a troca de genes referentes à matriz x_{ik}, então apenas a zona do cromossoma correspondente aos mesmos é considerada na escolha dos pontos e os genes dos valores das quantidades de produção são ignorados neste processo.

Depois de efetuada a troca de genes entre os indivíduos, o cromossoma pode ser sujeito a um processo de mutação. A ocorrência ou não de uma mutação está dependente do valor de probabilidade de mutação definido no início da execução do algoritmo. A mutação consiste em alterar aleatoriamente um dos genes da matriz x_{ik}, ou seja, alterar um produto na produção de uma máquina. Esta alteração é realizada tendo em conta quais os produtos que a máquina em questão pode produzir, de modo a que a mutação não inviabilize a solução.

Depois de completa a matriz x_{ik}, é ajustada a matriz q_{ik} para cada um dos novos indivíduos, tendo em conta as alterações causadas pela troca na matriz x_{ik}. Apenas são ajustadas as quantidades das produções referentes a produtos afetados no processo de troca de genes, por exemplo, se forem trocadas porções de genes onde estão representados

os produtos 3, 5 e 8, então apenas as quantidades das produções destes três produtos são ajustadas, de modo a garantir que restrição das quantidades das encomendas é cumprida, também tendo em conta os fatores de quantidades mínimas de produção por produto e incremento de quantidade de produção.

Com as matrizes x_{ik} e q_{ik} definidas, então é verificado se cada um dos novos indivíduos cumpre as restrições do modelo e pode ser considerado uma solução. Um individuo é adicionado à população, caso seja considera solução e se não existir já uma solução igual na nova população. Apenas depois de garantir que um individuo é adicionado à população é que é calculada a matriz de alocação temporal t_{ik} e o valor de *fitness* da solução. Optou-se por esta metodologia por uma questão de reduzir o tempo de execução do algoritmo, pois é desnecessário consumir tempo a calcular a matriz t_{ik} e o valor de *fitness* de um indivíduo que não vai ser adicionado à população.

Este processo de *crossover* é assim repetido iterativamente, até ser gerada uma nova população de soluções do tamanho pretendido, que corresponde a uma nova geração do algoritmo.

3.3.2.4 Critério de Paragem

O critério de paragem do algoritmo é composto por duas condições. Para que ocorra o término da execução basta que uma delas seja satisfeita. No final da execução, o resultado do algoritmo é a melhor solução encontrada nas várias gerações obtidas durante a execução.

A primeira condição de paragem é o limite no número de gerações. Este valor limite é definido à priori da execução e quando esse valor de gerações for atingido, é terminada a execução.

A outra condição de paragem é a de não melhora de solução durante um período de gerações, equivalente a um terço do número de gerações limite da primeira condição. Este condição visa determinar quando o algoritmo deve terminar a sua execução, com base na observação do comportamento da melhor solução ao longo das gerações. Quando a melhor solução não demonstra melhorias após um o número de gerações predefinido, o algoritmo termina. A escolha do valor de um terço do número de gerações limite, está relacionado com o facto de nos testes realizados com um limite de 100 gerações, ter-se observado que a melhor solução tendeu a estabilizar por volta das 60 gerações executadas.

Por exemplo, para Algoritmos Genéticos implementados através de soluções *Cloud*, este tipo de abordagem é bastante vantajosa, pois tenta reduzir o tempo de execução, sendo que estes serviços, por norma, são cobrados de acordo com o tempo de utilização dos mesmos.

3.3.3 Arquitetura e Estrutura de Dados

A implementação de um Algoritmo Genético envolve várias componentes e processos. A solução desenvolvida foi projetada de forma modelar, de acordo com a divisão entre entidade e respetivos processes na execução do algoritmo. Esta organização e separação

entre processos facilita a implementação e manutenção da solução. Na Figura 3.7 é apresentado o diagrama de classes do algoritmo, com os reptivos atributos e métodos.

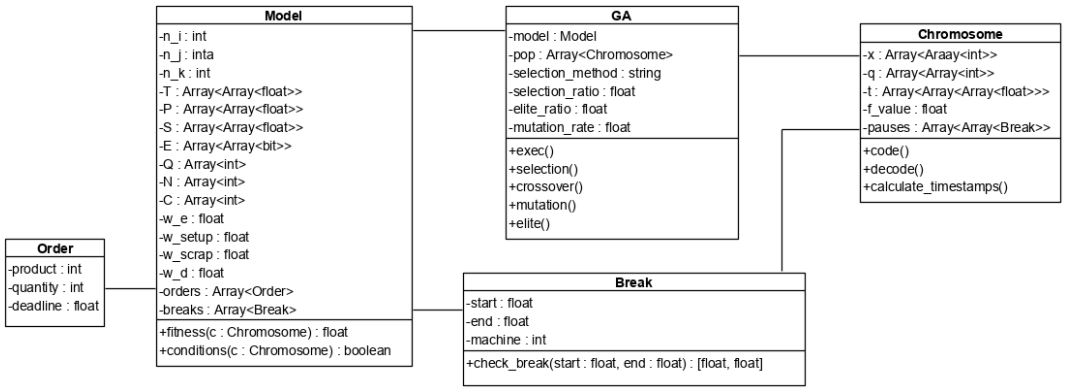

Figura 3.7: Diagrama de Classes da implementação.

A classe *GA* é a que representa o núcleo do Algoritmo Genético, composto pelos parâmetros e os processos de execução. Esta classe necessita de uma instância da classe *Model*, que representa o modelo do problema, através dos método que permitem o cálculo da função objetivo e verificação de restrições para uma determinada solução.

O cromossoma é representado pela classe *Chromosome* e a população de soluções é uma lista de instâncias desta classe. Cada encomenda gera uma instância da classe *Order* e a lista de encomendas é guardada na classe *Model*. As pausas na produção causadas por ausência de turno ou manutenções/limpezas previamente agendadas são representadas pela classe *Break*. Esta classe possui o método *check_break*, que para um certo intervalo de tempo verifica se existe conflito com a respetiva pausa. Caso exista conflito, é retornado um novo intervalo de tempo, que resolva o conflito mas mantenha a duração. Este método é o utilizado para verificar e resolver conflitos entre operações e pausas, quando é realizada a alocação temporal das produções.

3.3.4 Versões Alternativas

Depois de desenvolvida a versão base do Algoritmo Genético, foram testadas várias alterações, na tentativa de melhorar as soluções obtidas. Em seguida são apresentadas as diferentes versões implementadas do algoritmo, em relação à versão base já apresentada.

3.3.4.1 Algoritmo Genético Híbrido com Tabu Search

A otimização de soluções geradas por meio de algum tipo de algoritmo ou método, é um tema bastante recorrente no desenvolvimento de estratégias de resolução para problemas de otimização. A capacidade de aliar diferentes técnicas e métodos pode permitir obter melhores soluções do que apenas utilização singular de apenas uma metodologia [5].

Desta forma, pode ser vantajoso acrescentar ao algoritmo base desenvolvido, algum tipo de método de otimização para as soluções geradas. Estes métodos de otimização, por norma, focam-se numa procura local, dentro da vizinhança de uma solução. Por vezes, o difícil é definir a vizinhança de uma solução, pois este critério já depende do contexto do problema em si.

Posto isto, foi realizada uma versão híbrida do Algoritmo Genético, em que recorre simultaneamente a outro método de otimização de soluções. O método de otimização escolhido foi o *Tabu Search*. Este é um método de otimização que realiza uma procura no espaço de soluções possíveis, a partir de uma solução inicial, enquanto constrói uma "lista tabu"com o histórico das soluções já encontradas. Esta lista é utilizada para evitar verificar as mesmas soluções durante a procura [14, 18]. A lista possui um tamanho limite e sempre que esse limite é atingido. a solução mais antiga é removida da lista para acrescentar uma nova. A partir da solução inicial base é gerado um conjunto de soluções vizinhas. Se alguma das soluções vizinhas for melhor do que a solução base, então essa torna-se na nova base [14] e quando for satisfeito algum critério de paragem é retornada a melhor solução encontrada.

Deste modo, o *Tabu Search* é utilizado para otimizar, individualmente, cada uma das soluções da população inicial do Algoritmo Genético, através de uma pesquisa na vizinhança das mesmas. Para o contexto do problema de planeamento de produção apresentado, é considerada que uma solução está na vizinhança de outra, caso apenas uma das entradas da matriz x_{ik} seja diferente entre ambas. Isto faz com que o processo de gerar soluções vizinhas seja similar ao processo de mutação, ou seja, através do processo de mutação é sempre gerada uma solução vizinha da original.

Na implementação realizada do *Tabu Search* foi definido que o critério de paragem é o número de iterações e o tamanho da lista tabu é obtido pela divisão do número limite de iterações pelo número de soluções vizinhas geradas por iteração.

Um dos desafios presentes nesta abordagem é o tempo de execução total do algoritmo, pois este processo irá naturalmente aumentar o tempo necessário para obter a população inicial do Algoritmo Genético. Para não prejudicar em demasia a performance do algoritmo, é apenas executada uma iteração do *Tabu Search* com vizinhança de dez soluções, para cada solução da população inicialmente gerada.

Otimizar todas as soluções geradas ao longo das gerações do Algoritmo Genético, recorrendo ao *Tabu Search*, não se revelou uma solução viável devido ao grande aumento no tempo de execução. No entanto, compensa realizar a otimização da solução final do Algoritmo Genético. Como neste caso apenas será otimizada uma solução, então é possível aprofundar a pesquisa, sem aumentar em demasia o tempo de execução. Portanto, na tentativa de ainda otimizar a melhor solução encontrada pelo Algoritmo Genético, é aplicado o *Tabu Search*, com quinhentas iterações e uma vizinhança de dez soluções por iteração. A execução do Algoritmo Genético com *Tabu Search* está representada no diagrama da Figura 3.8.

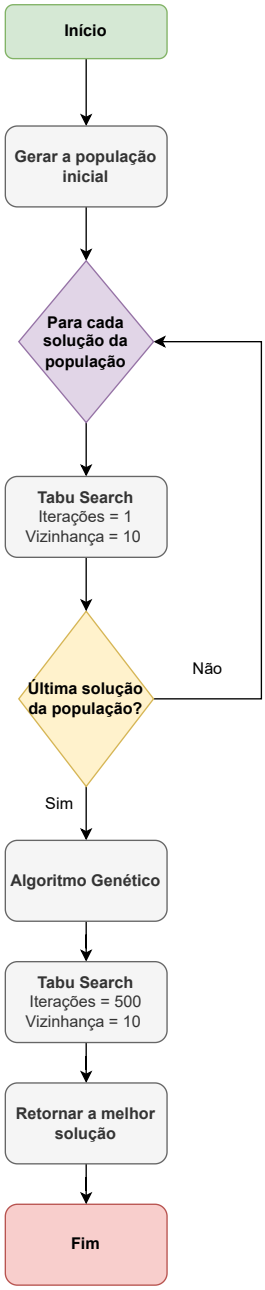

Figura 3.8: Diagrama de execução do Algoritmo Genético Híbrido com Tabu Search.

3.3.4.2 Algoritmo Genético Aleatório

A população inicial do Algoritmo Genético é gerada de forma aleatória. Neste caso, as soluções são geradas de forma aleatória, mas garantindo que são cumpridas as restrições do problema. O processo de gerar uma nova solução é o mesmo utilizado para gerar as soluções da população inicial.

A metodologia desta versão do algoritmo, consiste em gerar uma população totalmente nova a cada geração. Sendo assim, são ignorados os processos de Elite, Seleção, *Crossover* e Mutação, o que faz com que não exista relação entre gerações consecutivas.

Estas alterações fazem com que este método perca um pouco a essência de um Algoritmo Genético, no entanto, esta versão do algoritmo foi desenvolvida, principalmente, para comparar com a versão base e a híbrida. É importante entender se o Algoritmo Genético é mais eficaz e eficiente a resolver este problema de otimização do que uma metodologia baseada no processo de apenas gerar soluções novas de forma aleatória. A execução do Algoritmo Genético Aleatório está representada no diagrama da Figura 3.9.

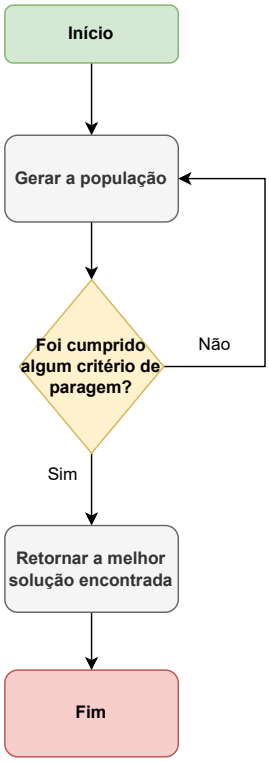

Figura 3.9: Diagrama de execução do Algoritmo Genético Aleatório.

3.3.4.3 Algoritmo Genético com *Bootstrapping*

Um dos fatores que mais influencia as soluções encontradas na execução de um Algoritmo Genético é a população inicial utilizada. Em teoria, quanto melhores forem as soluções presentes na população inicial, melhores serão as gerações seguintes.

Existem várias estratégias para realizar a otimização da população inicial, como por exemplo recorrer ao método *Tabu Search* (apresentado na Secção 3.3.4.1). A maioria dos métodos utilizados para otimizar a população inicial envolve a utilização de outros algoritmos auxiliares nesse processo. No entanto, é possível utilizar o próprio Algoritmo Genético para obter uma população inicial com melhores soluções. Este processo baseia-se na execução de um número de instâncias do Algoritmo Genético igual ao tamanho desejado para a população. A solução resultante de cada uma das execuções é adicionada à população inicial. Este processo pode ser referido como *Bootstraping* ou *Population Seeding*, e garante que, em teoria, a população inicial é constituída por melhores soluções do que se fosse gerada apenas por um processo aleatório [53].

Baseado neste principio, foi implementado uma versão do algoritmo que utiliza várias instâncias do Algoritmo Genético para gerar a população inicial. Se cada instância for executada com número de gerações e tamanho da população elevados, então o tempo de execução total será bastante elevado. Por este motivo, nas instâncias executadas para gerar a população inicial, são apenas utilizadas dez gerações e uma população com um tamanho equivalente a 10% do tamanho da população utilizada na execução principal do Algoritmo Genético. Esta metodologia, para gerar a população inicial, pode ser utilizado com qualquer uma das versões do Algoritmo Genético apresentadas anteriormente. A metodologia de execução do Algoritmo Genético com *Bootstraping* pode ser observada no diagrama da Figura 3.10.

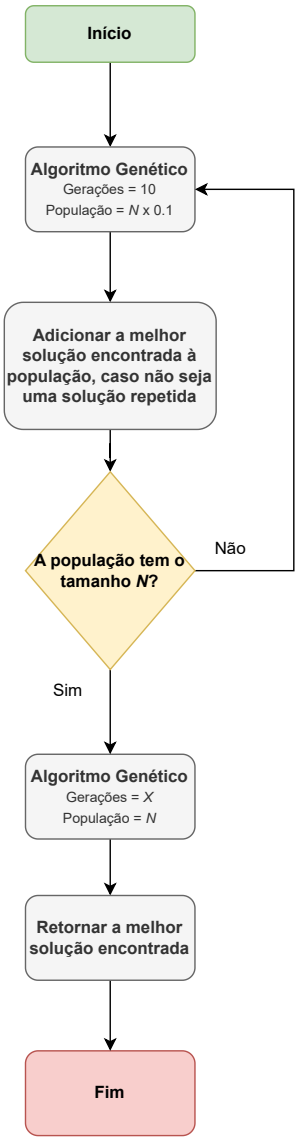

Figura 3.10: Diagrama de execução do Algoritmo Genético com *Bootstraping*.

4

Resultados

A fase de desenvolvimento e implementação resultou em diversas versões do Algoritmo Genético, capazes de solucionar o problema de otimização de planeamento de produção industrial, e alguns parâmetros passiveis de serem ajustados (método de seleção, taxa de mutação,...). Deste modo, é importante tentar determinar quais os valores destes parâmetros que melhor se ajustam ao problema em causa e sujeitar todas as versões desenvolvidas a diferentes cenários de teste, de modo a ser possível determinar qual a que melhor se adequa para o problema de produção industrial.

Depois de decididos quais os valores a usar nos diferentes parâmetros e qual a melhor versão algoritmo, então é importante aplicar o mesmo a um caso real e analisar os resultados obtidos.

Neste capitulo, são apresentados e analisados os resultados dos testes realizados em ambiente simulado e aplicação do algoritmo selecionado num caso real de produção industrial.

4.1 Testes em Ambiente Simulado

Os testes em ambiente simulado revelaram-se fulcrais no desenvolvimento do algoritmo, pois permitiram testar diversos cenários, de uma forma rápida e eficaz.

Através dos resultados obtidos, foi possível tomar decisões relevantes na implementação do algoritmo, ajustar valores de parâmetros e no fim comparar as diferentes versões desenvolvidas do algoritmo.

Nesta secção é apresentada a metodologia geral de testes realizadas, os resultados obtidos e a respetiva análise dos mesmos.

4.1.1 Metodologia de Teste

A metodologia de teste desempenha um papel crucial na validação e avaliação adequada do algoritmo.

Para começar, é importante definir claramente os objetivos dos testes e as métricas utilizadas para avaliar o desempenho do algoritmo. Isso inclui a escolha de indicativos de

performance, como o tempo de execução e a qualidade das soluções encontradas, através dos valores de *fitness*. Estes indicativos servem para realizar comparações entre diferentes decisões e versões, tornando possível analisar e interpretar os resultados, tendo por base critérios concretos e uniformes.

Além disso, a escolha dos cenários a testar desempenha um papel crucial na avaliação do algoritmo. Os cenários devem tentar abranger uma ampla gama de situações que o algoritmo pode encontrar num caso real. A diversidade dos cenários ajuda a verificar a robustez e capacidade de adaptação do algoritmo.

Para os testes a realizar, foram escolhidos três combinações diferentes de número de máquinas, produtos, ordens de produção e encomendas. Estes tentam representar casos industrias de pequena, média e média/elevada dimensão, respetivamente. Cada um dos casos foi pensado de modo a manter alguma coerência na relação entre os valores, por exemplo, fábricas com um maior número de máquinas e produtos tendem a receber mais encomendas. A escolha deste valores é sempre um processo relativo, visto que cada fábrica é um caso específico e não é possível arranjar um conjunto de valores que representasse todos. Na Tabela 4.1 são apresentados os três cenários de teste idealizados.

Cenário	**Máquinas (I)**	**Produtos (J)**	**Ordens (K)**	**Encomendas (E)**
1	3	5	5	100
2	10	20	15	200
3	30	50	20	350

Tabela 4.1: Cenários de Teste

Um dos desafios de realizar testes num ambiente simulado, está no processo de conseguir replicar casos reais através de dados gerados artificialmente. É necessário que estes dados simulem da melhor forma um caso real, caso contrário os resultados obtidos podem não traduzir fielmente o desempenho do algoritmo num caso real.

Posto isto, foi realizada uma análise para identificar quais os intervalos de valores *standard* associados a certos processos e recursos industriais, mais concretamente, na industria de molde por injeção de plástico.

- P_{ij} **(tempo de *setup*)** - valores inteiros gerados com distribuição uniforme entre 1000 e 10000 (segundos);
- S_{ij} **(porção de *scrap*)** - valores gerados com distribuição uniforme entre 0.01 e 0.06;
- T_{ij} **(tempo de ciclo/unidade)** - valores gerados com distribuição uniforme entre 20 e 35 (segundos);
- C_i **(consumo energético)** - valores gerados com distribuição uniforme entre 90 e 110 (kW/h);

- E_{ij} **(aptidão de produção por máquina)** - valores binários gerados de forma aleatória, mas garantindo que cada máquina é capaz de produzir pelo menos um produto e que cada produto pode ser produzido por pelo menos uma máquina;
- N_j **(limite de produções simultâneas por produto)** - valores inteiros gerados com distribuição uniforme entre 1 e 2;
- **Quantidade mínima por Produção** - 1000 unidades;
- **Incremento de Quantidade por Produção** - 500 unidades;
- **Pausas** - a fábrica pára a produção aos fins-de-semana e das 00h00 às 08h00 de Segunda-feira a Sexta-feira.
- **Encomendas** - para cada encomenda o produto é escolhido aleatoriamente e os valores de quantidades são aleatoriamente escolhidos do conjunto {1000, 2000, 3000, ..., 10000}. As datas de entrega também são geradas de forma aleatória, dentro do espaço temporal de 3 semanas a 3 meses, a contar desde a data de início definida para o planeamento.

Uma vez que os valores são gerados de maneira aleatória e o próprio funcionamento do algoritmo envolve diversos processos aleatórios, a realização de um único teste em cada cenário pode não ser suficiente para garantir resultados conclusivos e fiéis à realidade. Por essa razão, em cada cenário, são realizados dez testes separados, com novos valores gerados a cada execução e cem gerações. No final, é então realizada uma média dos resultados obtidos, com o propósito de atenuar e eliminar possíveis oscilações nos resultados, causadas por fatores não diretamente relacionados com a análise pretendida.

Todos os testes realizados foram executados numa máquina com um processador Intel i7-8750H (2.20GHz) e 16GB de memória RAM.

4.1.2 Configurações e Parâmetros

Na execução de um Algoritmo Genético existem diversos parâmetros configuráveis e que impactam diretamente a execução do mesmo e os resultados obtidos. Desta forma, é importante tentar ajustar da melhor forma este parâmetros para conseguir atingir a melhor performance possível do algoritmo a diferentes níveis.

Para tal, foi escolhido o cenário de teste de média dimensão ($I = 10$, $J = 20$, $K = 15$ e $E = 200$) para testar diferentes valores para os parâmetros. Para cada valor testado, foi executado o algoritmo para dez exemplos distintos do cenário de teste de média dimensão. No final, foi calculada a média dos tempos de execução e dos valores da função objetivo, obtidos nas dez execuções. Cada execução foi realizada durante cem gerações, com uma população de duzentos indivíduos. Apenas foi testado o cenário de média dimensão, devido ao elevado tempo total de execução dos testes que seria necessário para testar todos os cenários, pois para cada valor e cenário testados são executados 10 exemplos.

Nesta secção, são descritos os diversos testes realizados, com o intuito de determinar os valores mais adequados para cada um desses parâmetros. e analisados os respetivos resultados.

4.1.2.1 Porção e Método de Seleção

O processo de seleção possui dois parâmetros variáveis, que são definidos aquando da execução do Algoritmo Genético. Estes parâmetros são o método de seleção utilizado e a porção de indivíduos da população que são selecionados. Como ambos os parâmetros estão relacionados diretamente com o mesmo processo, então é relevante realizar uma análise conjunta dos dois.

Foram testados os dois métodos de seleção (Torneio e Roleta) e diferentes valores de porção de seleção, entre 0.3 a 0.7, com intervalos de 0.05.

Na Figura 4.1 é possível observar o gráfico com os resultados dos testes realizados, na comparação entre diferentes métodos e porções de seleção.

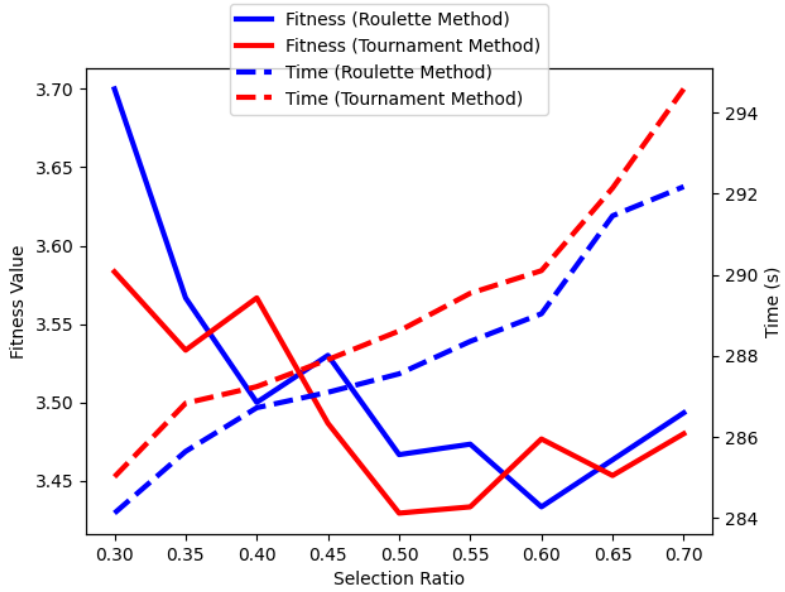

Figura 4.1: Resultados da Comparação entre Métodos e Valores da Porção de Seleção.

Ao analisar o gráfico, uma das primeiras conclusões que podemos retirar, é que o tempo de execução tende a aumentar à medida que o valor da porção de seleção aumenta, para ambos os métodos. Este fenómeno deve-se ao facto de quanto maior a porção de indivíduos a selecionar, mais tempo é gasto no processo de seleção, pois é necessário selecionar mais indivíduos. Ainda, no que toca ao tempo de execução, as diferenças registadas entre diferentes valores de porção e ambos os métodos de execução são pouco significativas, sendo que o Método da Roleta revelou-se ligeiramente mais eficiente.

Quanto aos valores obtidos para a função objetivo, estes também tendem a diminuir à medida que o valor de porção de seleção aumenta, mas não tão evidente e linear como o tempo de execução. Para ambos os métodos, os melhores resultados foram obtidos

para valores entre 0.5 e 0.65, sendo que neste intervalo, o Método do Torneio obteve maioritariamente melhores resultados.

Tendo em conta a análise realizada dos resultados obtidos, decidiu-se definir como método de seleção o Método de Torneio e um valor de porção de seleção de 0.5. O tempo de execução acabou por não ter peso na decisão, pois como já foi referido, as diferenças registadas são pouco significativas, e deste modo, o valor da função objetivo foi o principal critério de comparação.

4.1.2.2 Porção de Elite

Para selecionar a elite a cada geração, é primeiro necessário definir qual a porção de indivíduos são selecionados. É importante manter um balanço entre a continuidade dos melhores indivíduos entre gerações e a exploração de novas soluções, pertencentes espaço de possíveis soluções. Por este motivo, o valor de porção da elite não deve ser muito elevado, pois pode limitar a procura no espaço de soluções.

Foram testados valores de porção da elite, entre 0.05 a 0.5, com intervalos de 0.05. Na Figura 4.2 é possível observar o gráfico com os resultados dos testes realizados, na comparação entre diferentes valores de porção de elite.

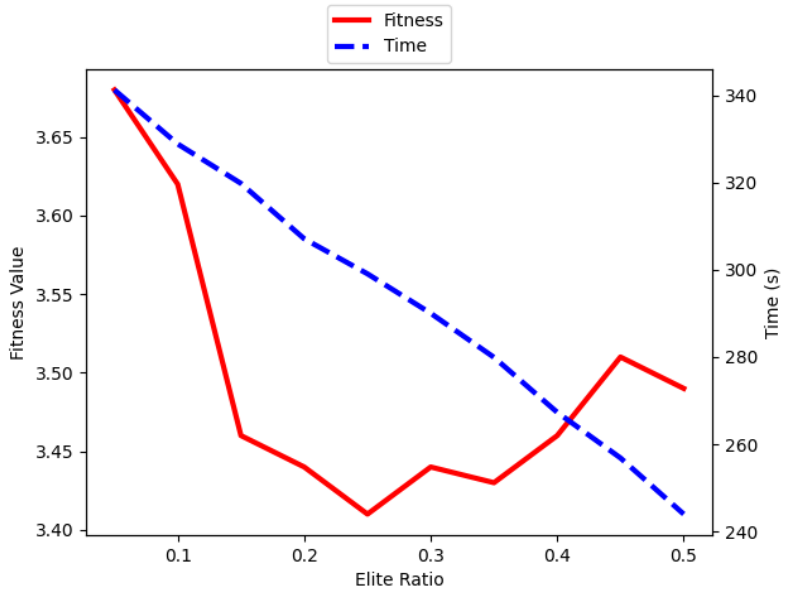

Figura 4.2: Resultados da Comparação entre Valores da Porção da Elite.

Ao analisar o gráfico, a primeira conclusão que conseguimos retirar, é que o tempo de execução tende a diminuir à medida que o valor da porção de elite aumenta. A diminuição do tempo de execução está relacionada com o maior número de indivíduos de elite que são automaticamente selecionados para a fase de reprodução e para a geração seguinte, diminuindo o número de indivíduos que necessitam ser selecionados pelo método de seleção e o número de novas soluções que necessitam ser geradas no processo de *crossover*. Quanto ao fenómeno da diminuição dos valores da função objetivo, até ao valor de 0.25,

deve estar relacionado com o aumento da qualidade dos indivíduos ao longo das gerações, visto que a elite é composta pelos melhores indivíduos da geração anterior. No entanto a partir de 0.3, o valor da função objetivo começa a subir gradualmente. Isto pode ser causado por existir uma menor variabilidade na população e reduzir a procura no espaço de soluções, visto que o conjunto de elite é maior e menos indivíduos são gerados na fase de reprodução.

Assim, tendo em conta a análise realizada dos resultados obtidos, decidiu-se definir como 0.25 o valor de porção de elite, pois é o valor que obteve melhor resultado a nível de valor da função objetivo e tempo de execução.

4.1.2.3 Probabilidade de Mutação

A probabilidade de mutação determina a frequência com que estas ocorrem nas soluções geradas através do processo de *crossover*. O processo de mutação serve então para introduzir mais um fator aleatório no algoritmo, de modo a tentar tornar a procura no espaço de soluções mais amplas e tentar evitar que o algoritmo caia num mínimo/máximo local. Assim como a questão da dimensão do grupo de elite, é crucial equilibrar a criação de novos indivíduos, exclusivamente, por meio do processo de *crossover* e a busca por novas soluções. Este aspeto garante que boas características tendam a ser mantidas ao longo das gerações, mas também permite explorar novas possibilidades.

Foram testados diferentes valores de probabilidade de mutação, entre 0.05 a 0.2, com intervalos de 0.05. Na Figura 4.3 é possível observar o gráfico com os resultados dos testes realizados, na comparação entre diferentes valores da probabilidade de mutação.

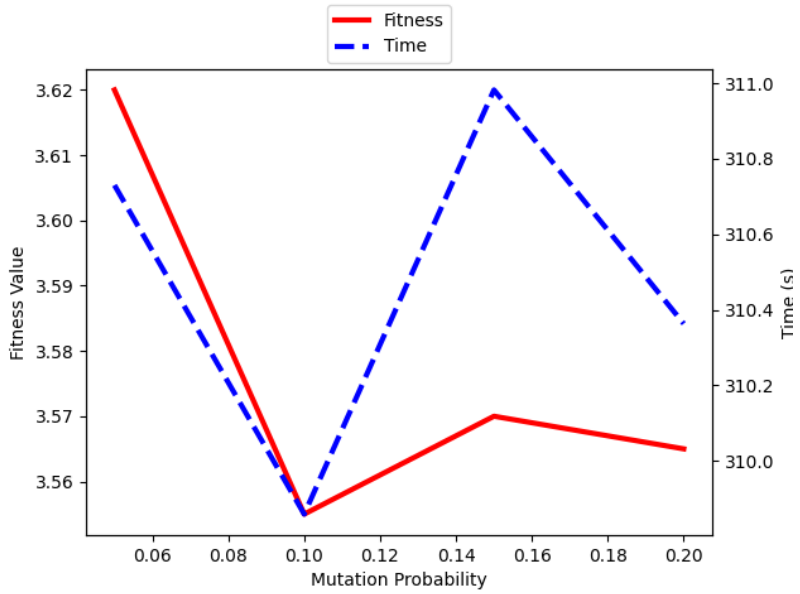

Figura 4.3: Resultados da Comparação entre Valores da Probabilidade de Mutação.

Ao analisar os resultados, notamos que as diferenças nos tempos de execução são pouco significativas, pelo que podemos descartar o tempo de execução como critério na

escolha do valor de probabilidade de mutação.

Quanto aos valores da função objetivo, este é um pouco complicado de analisar, pois os resultados não seguem uma tendência clara. Isto, pode dever-se ao facto de este ser um parâmetro associado a um processo com um grau de aleatoriedade. Mesmo assim, é possível verificar que para os valores de 0.1, 0.15 e 0.2 os resultados obtidos são geralmente melhores do que para o valor de 0.05, sendo que foi o valor de 0.1 que obteve o melhor valor de função objetivo, de entre os quatro.

Assim, tendo em conta a análise realizada dos resultados obtidos, decidiu-se definir como 0.1 o valor de probabilidade de mutação, pois é o valor que obteve melhor resultado a nível de valor da função objetivo.

4.1.3 Comparação entre Versões do Algoritmo

Da fase de desenvolvimento e implementação, resultaram três versões do Algoritmo Genético e a estratégia de *Bootstraping*, para a inicialização da população. Cada uma das versões tem as suas especificidades e é importante submeter as mesmas a um conjunto de testes, de forma a determinar qual das versões melhor se adequa ao problema de planeamento de produção industrial.

Desta forma, para a realização de todos os testes apresentados nesta secção, foram utilizados os três cenários de teste já apresentados e dez exemplos, com valores distintos, para cada um dos cenários. Para cada cenário, foi calculada a média dos diferentes valores obtidos para os dez exemplos. Cada execução foi realizada uma população de duzentos indivíduos e com cem gerações, sendo que os resultados foram recolhidos de dez em dez gerações, ao longo da execução. Os pesos utilizados no cálculo da função objetivo foram os seguintes : $W_e = 0.1$, $W_scrap = 0.1$, $W_setup = 0.2$ e $W_d = 0.6$. Estes valores foram definidos depois de realizada uma análise, com a ajuda da *Muvu Technologies*, sobre quais fatores, os fabricantes dão mais relevância a nível de planeamento. Chegou-se à conclusão que o cumprimento de prazos de entrega, seguido da diminuição do tempo de *setup*, são, de modo geral, os fatores mais relevantes para os fabricantes, resultando assim na distribuição de pesos utilizada.

Nesta secção, são descritos os diversos testes realizados, primeiramente com o intuito de comparar as diferentes versões desenvolvidas do Algoritmo Genético e posteriormente para analisar o impacto da estratégia de *Bootstraping*, a nível de resultados e tempo de execução.

4.1.3.1 AG vs AG Híbrido vs AG Aleatório

Com o objetivo de comparar e analisar as três versões do Algoritmo Genético, estes foram submetidos a um conjunto de testes de execução. O intuito destes testes é analisar a performance da cada uma das versões do Algoritmo, a nível de tempo de execução e valores obtidos para a função objetivo e os seus componentes (datas de entrega, consumo energético, tempo de *setup* e quantidade de *scrap*.

As três versões do algoritmo foram parametrizadas de igual forma e sujeitas aos mesmos exemplos, de modo a existir um termo de comparação e análise dos resultados obtidos. Para cada exemplo, a população inicial utilizada em cada uma das versões foi a mesma, sabendo que no caso do Algoritmo Genético Híbrido, esta tenha sido posteriormente otimizada, devido à natureza da metodologia desta versão.

Na Figura 4.4 é possível observar os três gráficos com os resultados obtidos, para cada um dos cenários testados, com a comparação entre as diferentes versões do algoritmo, a nível de valor da função objetivo e tempo de execução.

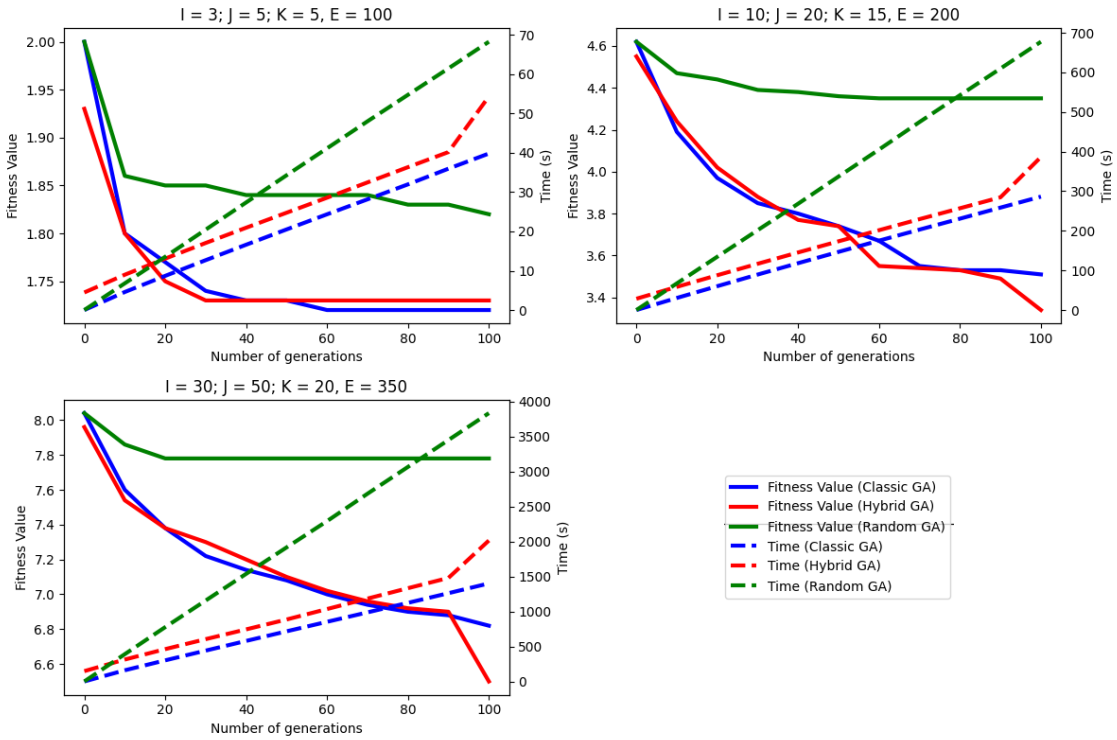

Figura 4.4: AG vs AG Híbrido vs AG Aleatório - *Fitness* & Tempo de Execução

Ao analisar os resultados obtidos, podemos notar, que para todos os cenários, os tempos de execução de cada versão seguem a mesma tendência e que o tempo de execução aumenta de forma generalizada à medida que aumenta a complexidade dos cenários.

O Algoritmo Genético Aleatório é o que revela maior tempo de execução, como seria expectável, pois gerar uma população nova a cada geração é um processo mais demorado do que o conjunto de processos utilizados tradicionalmente num Algoritmo Genético (elite, seleção, *crossover* e mutação).

Quanto ao Algoritmo Genético e o Algoritmo Genético Híbrido, estes possuem um padrão similar a nível de tempos de execução, pois os processos utilizados durante grande parte da execução são idênticos, sendo que a diferença está no início e fim da execução. O tempo de execução acrescido, que a versão Híbrida apresenta, na fase inicial, deve-se ao tempo despendido pela otimização da população inicial através da utilização do *Tabu Search*. O aumento registado no fim da execução também deve-se à otimização da solução

obtida, por meio do *Tabu Search*, mas neste caso é um aumento mais acentuado, pois é feita uma procura com mais profundidade, o que leva a um maior tempo despendido neste processo. Este aumento no tempo de execução é apenas registado na última geração do Algoritmo Genético Híbrido, apesar dos resultados no gráfico darem a entender que este ocorre a partir da geração noventa. Isto deve-se ao facto, de que os dados foram recolhidos com intervalos de dez gerações.

A nível de valores da função objetivo obtidos, estes diminuem ao longo das gerações, como era expectável devido à metodologia do algoritmo. O Algoritmo Genético Aleatório é a versão que obteve piores resultados, com uma diferença significativa para as restantes versões. Podemos assim concluir, que a metodologia dos Algoritmos Genéticos é significativamente mais eficaz e eficiente, na resolução deste tipo de problemas de otimização, do que simplesmente gerar soluções de forma aleatória.

Em relação, aos valores da função objetivo obtidos para o Algoritmo Genético e o Algoritmo Genético Híbrido, para os três cenários estas são similares ao longo das gerações. Para o cenário de pequena dimensão, o Algoritmo Genético conseguiu obter melhores resultados e a otimização, levada a cabo pelo *Tabu Search* no Algoritmo Genético Híbrido, não apresentou melhorias nos resultados. No entanto, nos restantes cenários essa otimização revelou-se vantajosa visto que existe uma melhoria significativa das soluções obtidas na última geração da execução.

Para a versão Híbrida, apesar de a população inicial ser otimizada, recorrendo ao *Tabu Search*, esta melhoria da população inicial não se traduziu numa melhoria significativa dos resultados ao longo das gerações, visto que os valores obtidos pelo Algoritmo Genético são similares. Só a otimização realizada na solução final é que se traduziu numa melhoria significativa.

Para perceber melhor o impacto dos resultados a nível de planeamento, é importante realizar uma análise dos diferentes fatores que compõem a função objetivo. Analisar valores concretos de consumo energético total, tempo de *setup*, quantidade de *scrap* e falhas nos prazos de entrega, e não analisar a nível de valor de cada componente na função objetivo, pois isso apenas seria uma decomposição dos valores já apresentado e não permite ter uma interpretação do impacto prático, a nível dos planeamentos gerados.

Cenário de Teste de Pequena Dimensão ($I = 3$; $J = 5$; $K = 5$, $E = 100$)

Na Figura 4.5 são apresentados os resultados obtidos, referentes às diferentes fatores que compõem a função objetivo, para os testes realizados com o cenário de teste de pequena dimensão. É possível constatar, que o Algoritmo Genético Híbrido apresenta melhores resultados para todos os componentes, exceto para o cumprimento dos prazos de entrega, em média, não cumprindo cerca de duas datas de entrega. Tendo em conta que o cumprimento dos prazos de entrega foi o fator com o maior peso nos testes, então mesmo obtendo os melhores resultados nos restantes fatores, o Algoritmo Genético Híbrido

acabou por obter pior valor final de função objetivo do que o Algoritmo Genético.

O mesmo acontece, de forma semelhante, no caso do Algoritmo Genético Aleatório, que obtém valores semelhantes ao Algoritmo Genético em todos os fatores, exceto no cumprimento de prazos de entrega, e por isso, no final acaba por obter um pior valor de função objetivo.

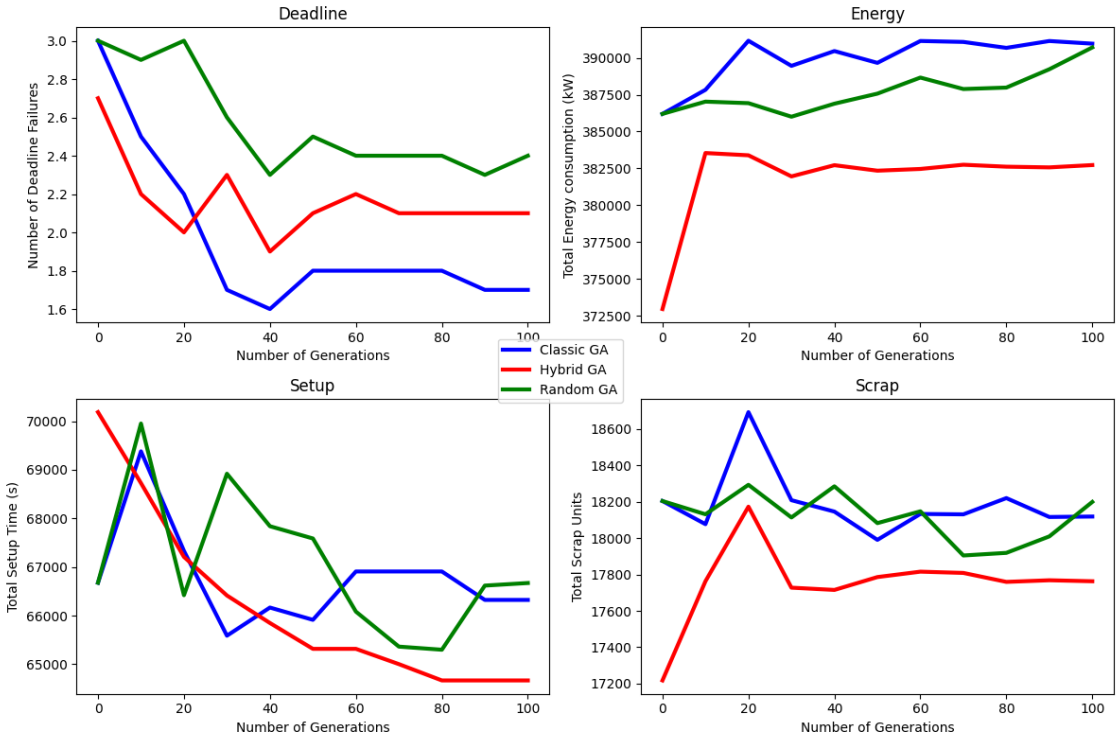

Figura 4.5: AG vs AG Híbrido vs AG Aleatório - ($I = 3$; $J = 5$; $K = 5$, $E = 100$)

Cenário de Teste de Média Dimensão ($I = 10$; $J = 20$; $K = 15$, $E = 200$)

Na Figura 4.6 é possível observar os valores obtidos, referentes às diferentes fatores que compõem a função objetivo, para os testes realizados com o cenário de teste de média dimensão.

Neste caso, o Algoritmo Genético Aleatório obteve, significativamente, os piores resultados em todos os fatores, exceto na quantidade de *Scrap*. E de forma natural, este aspeto traduz-se em piores valores obtidos para a função objetivo, como é possível observar na Figura 4.4.

Quanto à versão clássica e híbrida do algoritmo, ocorre o oposto do teste anterior. Neste caso, o Algoritmo Genético Híbrido obteve melhor valor final de função objetivo, pois também obteve melhor resultado a nível de cumprimento de prazos de entrega, sendo que este fator teve o maior peso.

Em relação à otimização, realizada pelo método de *Tabu Search*, esta levou a uma

redução no incumprimento de prazos de entrega e consumo energético, e a um aumento no tempo de *Setup* e quantidade de *Scrap*, que no todo traduziu-se numa diminuição do valor da função objetivo, ou seja, a solução foi otimizada.

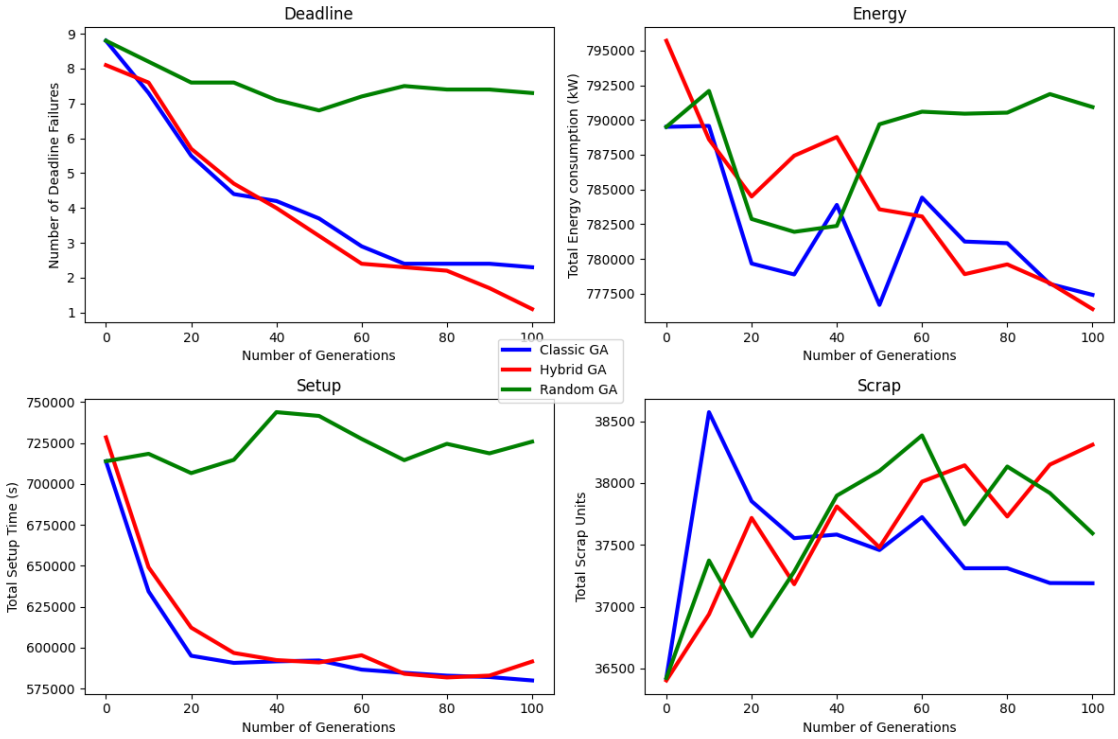

Figura 4.6: AG vs AG Híbrido vs AG Aleatório - ($I = 10$; $J = 20$; $K = 15$, $E = 200$)

Cenário de Teste de Média/Elevada Dimensão ($I = 30$; $J = 50$; $K = 20$, $E = 350$)

Na Figura 4.7 é possível observar os valores obtidos, referentes aos diferentes fatores que compõem a função objetivo, para os testes realizados com o cenário de teste de média/elevada dimensão.

A análise dos resultados deste conjunto de testes, acaba por ser semelhante e ir de encontro à análise realizada dos resultados obtidos nos testes com o cenário de média dimensão. Neste caso, o Algoritmo Genético Híbrido obteve melhores resultados em todos os fatores e ainda é mais notória a melhoria obtida através da otimização da melhor solução, com o método de *Tabu Search*. Esta otimização reduziu significativamente o valor de todos os fatores, com exceção da quantidade de *Scrap*, onde se registou um aumento. Mesmo com este aumento, devido à distribuição dos pesos utilizada, ocorreu uma redução do valor da função objetivo.

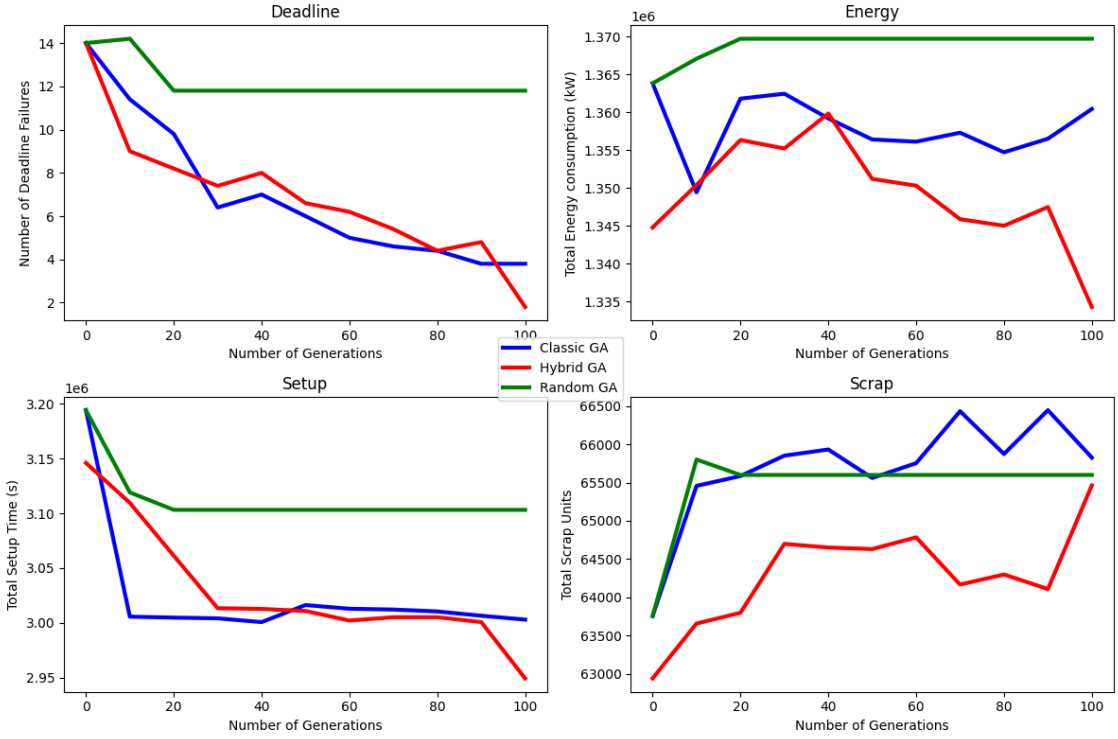

Figura 4.7: AG vs AG Híbrido vs AG Aleatório - ($I = 30$; $J = 50$; $K = 20$, $E = 350$)

Tendo em conta os resultados obtidos nos diversos testes e a análise dos mesmos, conclui-se que o Algoritmo Genético Híbrido apresenta, de modo geral, melhores resultados que as restantes versões. O uso do método de *Tabu Search*, para realizar a otimização da melhor solução encontrada pelo Algoritmo Genético, demonstrou-se uma técnica viável, devido às melhorias obtidas, a nível de valor da função objetivo. O aumento do tempo de execução causado por este processo de otimização é compensado pela melhoria obtida a nível de qualidade da solução, fazendo desta versão Híbrida, a que foi considerada mais apta para resolver este problema de planeamento de produção industrial.

4.1.3.2 *Bootstrapping*

Concluídos os testes das três versões do algoritmo, foi necessário testar o método de *Bootstrapping*, utilizado na inicialização da população inicial.

Este método, descrito na secção 3.3.4.3, baseia-se em utilizar múltiplas instâncias do Algoritmo Genético para gerar as soluções que compõem a população inicial. Com base nos resultados obtidos nos testes apresentados na secção 4.1.3.1, decidiu-se utilizar o Algoritmo Genético e o Algoritmo Genético Híbrido em conjunto nesta abordagem. Definiu-se que a versão base do Algoritmo Genético seria a escolhida para executar as múltiplas instâncias, para formar a população inicial e o Algoritmo Genético Híbrido seria utilizado na execução principal. Esta decisão está relacionada com o facto de o Algoritmo Genético Híbrido apresentar melhores resultados, mas um maior tempo de execução, que iria aumentar significativamente o tempo necessário para gerar a população inicial.

Devido a este fator, optou-se por utilizar o Algoritmo Genético no processo de gerar a população inicial.

O conjunto de soluções, obtido através deste processo, representa a população inicial utilizada pelo Algoritmo Genético Híbrido com o método de *Tabu Search*. Já no âmbito da execução do Algoritmo Genético Híbrido, a população inicial gerada através das múltiplas instâncias do Algoritmo Genético, é sujeita a um processo de otimização com o método de *Tabu Search*. No final, a melhor solução encontrada também é submetida ao mesmo processo de otimização com o método de *Tabu Search*, mas com uma procura mais alargada. Na Figura 4.8 é apresentado o diagrama com os diferentes processos e execução desta estratégia de *Bootstrapping*, tendo já em conta as decisões tomadas, baseadas nos resultados dos testes anteriormente apresentados.

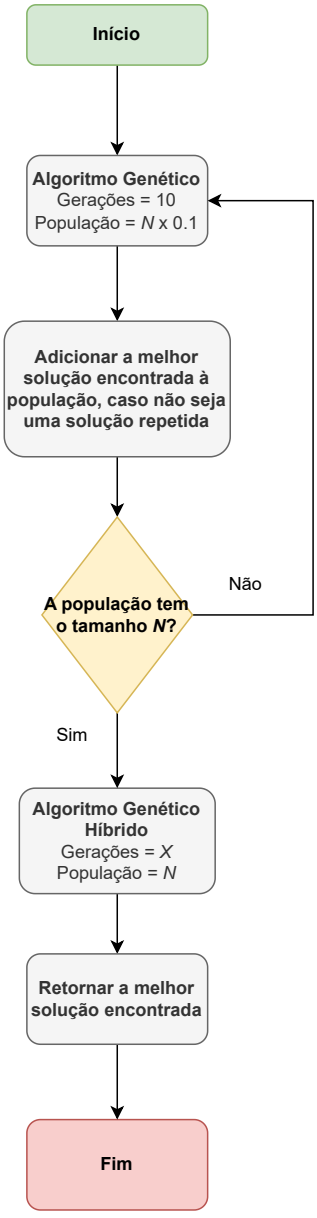

Figura 4.8: Diagrama de Execução do AG com *Bootstrapping* Testado.

Os testes foram realizados com o intuito de realizar uma comparação entre o método de *Bootstrapping* e o Algoritmo Genético Híbrido, visto que esta foi considerada a melhor das três versões do algoritmo, tendo em conta os testes realizados anteriormente. Na Figura 4.9 é possível observar os resultados dos testes realizados, com a comparação entre o Algoritmo Genético Híbrido e o Algoritmo com o método de *Bootstrapping*.

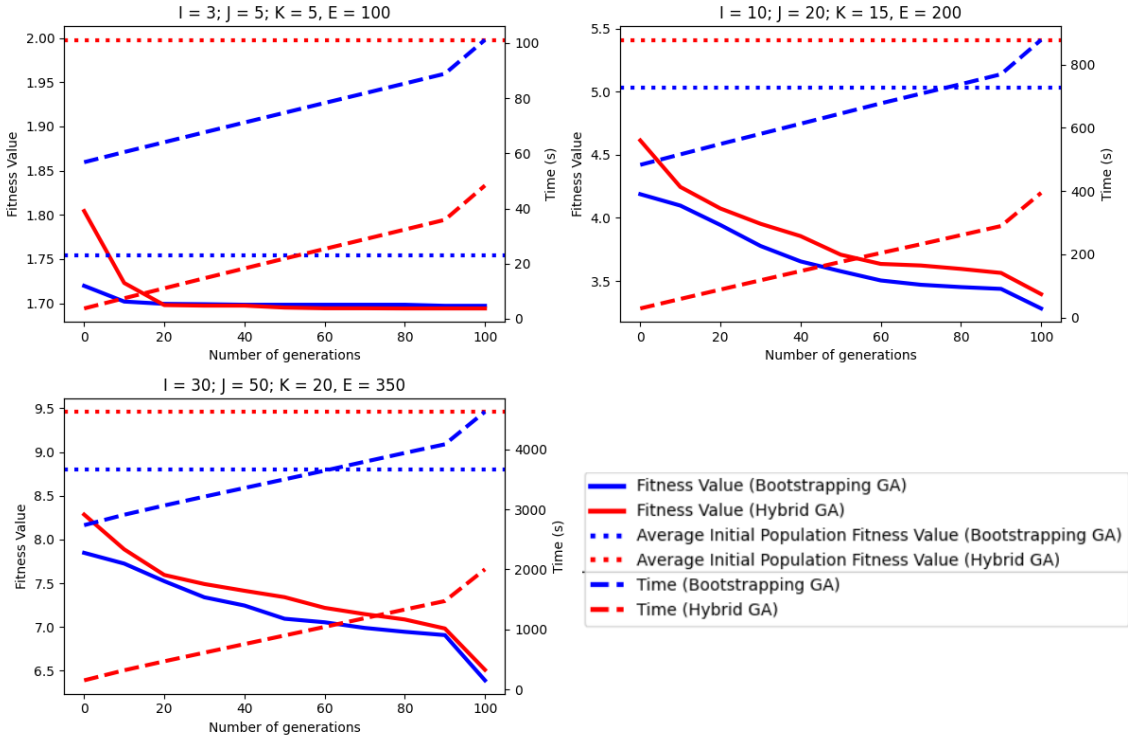

Figura 4.9: AG Híbrido vs AG com *Bootstrapping* - *Fitness* Tempo de Execução

Ao analisar os resultados obtidos, podemos notar, que para todos os cenários, os tempos de execução para ambos os casos seguem a mesma tendência e que o tempo de execução do algoritmo com o método de *Bootstrapping* é sempre mais que o dobro do tempo de execução do Algoritmo Genético. Os picos de tempo registados no final, são referentes à otimização da melhor solução encontrada, através do método de *Tabu Search*. Como os resultados foram recolhidos com intervalos de dez gerações, o gráfico dá a entender que esse pico do tempo de execução ocorre a partir da geração noventa, no entanto, este apenas ocorre na geração cem, a última geração. O mesmo acontece com os valores da função objetivo e a justificação para tal é idêntica.

Um fator que se revelou importante de analisar é a qualidade da população inicial, a nível de valor da função objetivo dos indivíduos que a compõem. Esta análise tem o objetivo de perceber o real impacto do método de *Bootstrapping* na inicialização da população. Foram então, calculadas as médias dos valores da função objetivo das populações iniciais geradas, para ambos os casos, respetivamente. Constatou-se que, para todos os cenários, o método de *Bootstrapping* obteve uma melhoria significativa em comparação com o Algoritmo Genético Híbrido, no que toca ao valor médio da população inicial. A

análise destes valores permitiu confirmar o impacto positivo esperado, que o método de *Bootstrapping* teve na melhoria da qualidade das soluções da população inicial.

Quanto ao valor da função objetivo das melhores soluções, ao longo das gerações, este tende a diminuir gradualmente, com exceção no cenário de pequena dimensão, em que ambos os casos atingiram um valor estável, aproximadamente a partir da vigésima geração. O algoritmo com o método de *Bootstrapping* obteve, de forma geral em todos os cenários, melhores soluções, com exceção do cenário de pequena dimensão. Porém neste caso, a diferença entre ambos é pouco significativa.

De um modo geral, podemos concluir que o método de *Bootstrapping* é eficaz na inicialização da população inicial, na medida em que permite gerar populações geralmente compostas por melhores soluções. No entanto, é um método pouco eficiente, em relação ao Algoritmo Híbrido, visto que o tempo de execução é sensivelmente o dobro que este e as soluções obtidas no final da execução não são significativamente melhores, a ponto de compensarem um aumento tão acentuado no tempo de execução. Desta forma, ao realizar um balanço entre soluções finais obtidas e tempo de execução, decidiu-se manter a escolha no Algoritmo Genético Híbrido, como a solução desenvolvida mais capaz para resolver o problema de planeamento de produção abordado.

4.2 Validação em Caso Real

Até agora, todos os resultados obtidos resultaram de testes realizados em ambiente simulado, com dados gerados de forma aleatória, mas com a tentativa que estes fossem o mais representativos possíveis da realidade.

A validação da solução, recorrendo a um caso real, é fundamental no desenvolvimento deste tipo de soluções, visto que são desenvolvidas para mais tardem poderem ser utilizadas em ambiente real de produção.

Nesta secção é apresentado o caso real utilizada neste processo de validação, a metodologia utilizada no mesmo e os resultados obtidos.

4.2.1 Enquadramento e Metodologia

O caso de estudo real, utilizado para validar a solução desenvolvida, corresponde a um planeamento de produção de uma fábrica da área da produção de produtos de plástico, através de molde por injeção. Esta fábrica é cliente da *Muvu Technologies* e os dados utilizados necessários, para realizar este teste da validação, foram obtidos através do sistema *RAILES*.

O planeamento corresponde a produções levadas a cabo na fábrica entre o final de Janeiro de 2023 e inicio de Junho de 2023. A fábrica possui 17 máquinas e no intervalo de tempo definido, produziu 130 tipos de produto diferentes, repartidos por diferentes produções, correspondentes às 190 encomendas realizadas. Cada encomenda possui uma

data de entrega e uma quantidade de produto a entregar. Os valores de quantidade neste conjunto de encomendas variam entre 100 a 1008000 unidades de produto a entregar.

A fábrica, utilizada neste caso de estudo, funciona através de um regime de três turnos diários durante a semana, com uma duração de oito horas cada, sendo que ao fim-de-semana e feriados a fábrica não se encontra em funcionamento.

Ao realizar esta validação da solução desenvolvida, num caso real, o objetivo é comparar o planeamento real que foi executado pela fábrica com o planeamento gerado pelo algoritmo, e verificar se existe, ou não, uma melhoria. Para tal, é necessário calcular o valor da função objetivo para o planeamento real e o respetivo consumo energético, tempo de *Setup*, quantidade de *Scrap* e incumprimento dos prazos de entrega das encomendas.

Neste processo de validação, foi utilizado o Algoritmo Genético Híbrido, pois mediante os resultados dos testes com dados simulados, foi considerada a melhor solução desenvolvida.

4.2.2 Planeamento Real vs Planeamento Gerado

Para ser possível comparar os resultados obtidos pelo Algoritmo Genético Híbrido com o planeamento executado pela fábrica, é necessário traduzir o planeamento real em valores objetivos. O valor da função objetivo está sempre dependente dos pesos atribuídos às diferentes componentes, aquando do cálculo da mesma, no entanto, valores totais como o consumo energético, o tempo de *Setup*, a quantidade de *Scrap* e o incumprimento dos prazos de entrega das encomendas de uma certa solução, estão apenas diretamente ligados ao planeamento e não variam consoante possíveis alterações efetuadas nos pesos.

Desta forma, foram calculados os valores totais das componentes para o planeamento real, tendo em conta os dados recolhidos na plataforma *RAILES*:

- Foram cumpridos todos os prazos para as encomendas;
- O consumo energético total foi de **294465.75 kW**;
- O tempo total de *Setup* despendido foi igual a **9030 minutos**;
- A quantidade total de *Scrap* foi igual **177841 unidades**.

Depois de obtidos os valores referentes ao planeamento real, foram testadas várias execuções com o algoritmo, de modo a obter resultados para combinações de pesos diferentes na função objetivo. As combinações de pesos testadas foram definidas na tentativa de tentar abranger diferentes cenários, no que toca às prioridades da fábrica a nível de construção do planeamento.

Cada execução foi realizada durante cem gerações com uma população de duzentos indivíduos, uma porção de seleção de 0.5, o método de seleção de torneio, uma porção de elite de 0.25 e uma probabilidade de mutação de 0.1. O tempo de execução do algoritmo foi similar nas execuções realizadas, para as diferentes combinações de pesos, sendo que variou entre 668.5 segundos e 707.6 segundos.

Na Tabela 4.2 são apresentados os resultados obtidos nos testes realizados. A cor verde nos valores indica uma discrepância positiva em relação ao valor real, a cor vermelha denota uma discrepância negativa, a cor laranja sinaliza uma discrepância negativa, embora esta permaneça próxima do valor original. Por fim, a cor amarela é indicativa de que os valores permaneceram inalterados.

	Wd = 0.6 *Wscrap = 0.1* *We = 0.1* *Wsetup = 0.2*	*Wd = 0.2* *Wscrap = 0.5* *We = 0.1* *Wsetup = 0.2*	*Wd = 0.1* *Wscrap = 0.6* *We = 0.1* *Wsetup = 0.2*	*Wd = 0.4* *Wscrap = 0.3* *We = 0.1* *Wsetup = 0.2*	*Wd = 0.1* *Wscrap = 0.2* *We = 0.5* *Wsetup = 0.2*	*Wd = 0.2* *Wscrap = 0.2* *We = 0.1* *Wsetup = 0.5*
Valor de FO do Planeamento Real	2.959	3.836	4.055	3.397	5.366	6.468
Valor de FO do Planeamento Gerado	2.764	3.618	3.828	3.211	4.993	6.206
Número de Incumprimentos de Prazos de Entrega	0	0	3	0	0	2
Tempo Total de Atraso nos Prazos de Entrega	0	0	378h 33m	0	0	248h 14m
Quantidade de Scrap Total (unidades)	180933	176235	175312	176342	176731	180771
Consumo Energético Total (kW)	294843.74	294604.23	294608.50	294719.97	294471.83	294800.70
Tempo Setup Total (minutos)	8120	8350	8480	8395	8125	7975

Tabela 4.2: Resultados do Teste de Validação do Caso Real

Ao analisar os resultados, podemos verificar que para todos os casos existiu uma melhoria a nível de valor da função objetivo, em relação ao valor do planeamento real. Esta melhoria é logo à partida um aspeto positivo, pois permite concluir que o algoritmo, mesmo aplicado a um caso real, é capaz de produzir planeamentos otimizados e com melhorias em relação ao planeamento real executado.

É importante ter em conta que valores iguais para um certo peso, não garantem os mesmos resultados para esse fator. A distribuição dos restantes pesos acaba por naturalmente influenciar os resultados obtidos.

A nível do incumprimento de prazos de entrega das encomendas, como no planeamento real todas as encomendas eram satisfeitas dentro do prazo, então não seria possível melhorar estes aspeto e apenas igualar o mesmo. Apenas em duas combinações de pesos testadas é que não o planeamento gerado não cumpre todos os prazos de entrega, porém nesses casos o peso atribuido ao cumprimento de *deadline* é baixo, o que pode justificar estes resultados.

Quanto à quantidade de *Scrap*, registou-se uma melhoria na maioria dos casos testados. Nos dois casos em que o valor da quantidade de *Scrap* total piorou, foi atribuído um peso mais baixo a este fator, o que, à partida, pode justificar este aumento.

Em termos de consumo energético, este foi o aspeto com os resultados menos favoráveis, já que não se registou melhorias em nenhum dos casos. No entanto, os valores registados não mostram uma diferença substancial em relação aos valores do planeamento original, e, no caso um peso de 0.5, foi possível obter um valor perto do planeamento real.

A nível de tempo de *Setup*, os resultados obtidos em relação ao obtido no planeamento real, foram significativamente melhores em todos os casos, com melhorias entre os 550 e 1055 minutos.

Comparando todos os diferentes casos testados, podemos concluir que a combinação de pesos que obteve melhores resultados, de modo geral, foi $W_d = 0.1$, $W_scrap = 0.2$, $W_e = 0.5$ e $W_setup = 0.2$. No entanto, a decisão de qual a melhor combinação de pesos

está sempre relacionada com as necessidades e objetivos do fabricante, sendo que não há uma combinação "ideal".

Na Figura 4.10 é apresentado o diagrama de Gantt do planeamento real e do planeamento gerado com os pesos $W_d = 0.1$, $W_scrap = 0.2$, $W_e = 0.5$ e $W_setup = 0.2$. Os eventos correspondem às produções alocadas ao longo do tempo nas diversas máquinas e as barras verticais cinzentas correspondem às pausas da fábrica nesse período. Devido a limitações impostas pela biblioteca utilizada, as cores nem sempre correspondem a produções do mesmo produto em ambos os diagramas e existem cores repetidas para produtos diferentes, visto que neste caso existem mais produtos do que cores disponibilizadas.

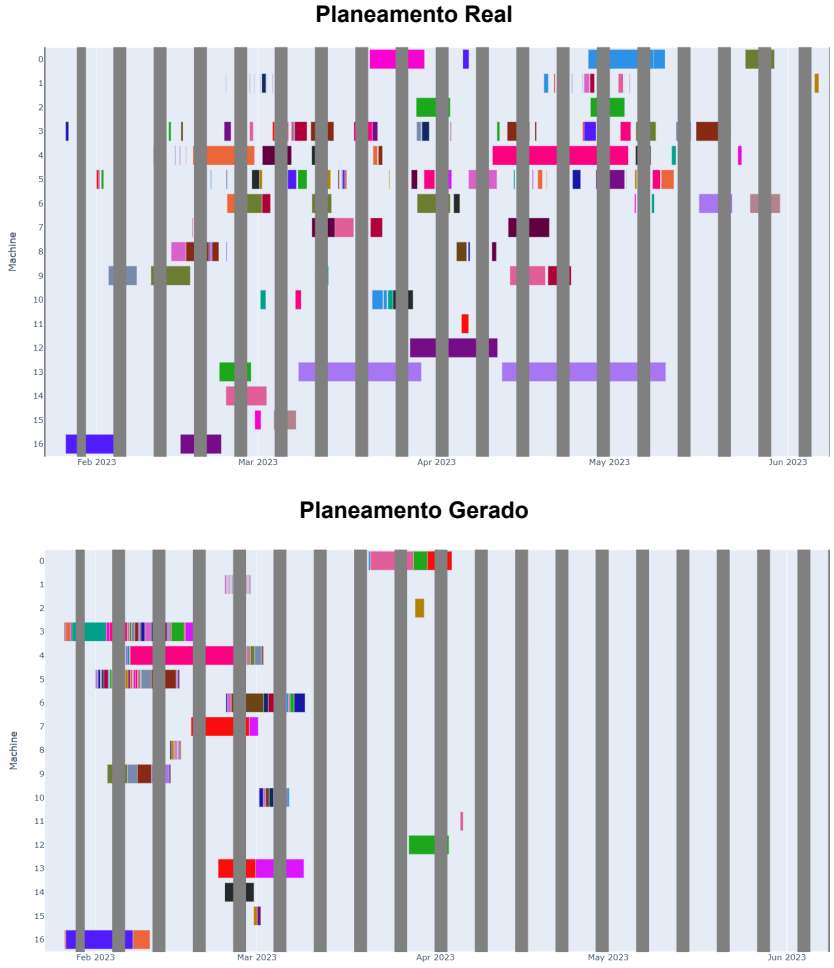

Figura 4.10: Gantt do Planeamento Real Vs Planeamento Gerado.

Ao analisar ambos os diagramas, a principal conclusão que é possível retirar, é que o planeamento gerado consegue satisfazer o mesmo conjunto de encomendas em aproximadamente metade do tempo que o planeamento real. Esta melhoria é bastante significativa, pois o facto de conseguir produzir o mesmo em menos tempo leva a uma maior disponibilidade de máquina, que pode ser capitalizada através do aumento do volume de negócios da fábrica, sem necessitar de recursos adicionais.

5

Conclusão

A evolução tecnológica, notada de forma evidente ao longo dos anos, tem permitido desenvolver soluções tecnológicas em diferentes áreas, como é o caso da área industrial. As *smart factories* são o futuro do mundo industrial, pois conseguem extrair as vantagens provenientes do uso de soluções tecnológicas atuais e assim melhorarem em muito os índices e qualidade de produção, em relação às fábricas convencionais.

A conectividade entre processos, recursos e componentes da fábrica, aliados a uma recolha de dados de produção em tempo real, levam a que seja possível implementar soluções de apoio à decisão, úteis para a gestão de topo da fábrica.

As ferramentas de planeamento de produção inteligente são um dos tipos de soluções de apoio à decisão, que só são possíveis de implementar devido à digitalização de processos e informação da fábricas, característicos da Indústria 4.0.

Sendo que o planeamento de produção é um dos aspetos fundamentais na gestão de uma fábrica, este tipo de ferramentas são de utilidade extrema e cada vez mais existe uma procura por este tipo de soluções, fazendo com que este seja um tema relevante e atual.

Deste modo, o algoritmo de planeamento desenvolvido, aplicado em ambiente real pode ter um impacto positivo significativo na gestão de produção da fábrica.

Aquando da conclusão do trabalho, é importante realizar um balanço final e perspetivar possíveis continuações do mesmo no futuro. Neste capítulo são apresentadas as conclusões finais relativas ao trabalho realizado e algumas ideias de trabalho futuro a realizar nesta área.

5.1 Conclusões

O trabalho desenvolvido pode ser divido em diferentes fases, em que cada uma teve um papel fundamental no resultado final obtido.

A análise realizada do problema de planeamento de produção e a identificação dos diferentes requisitos e necessidades, associadas ao mesmo, permitiram adquirir uma visão profunda e conhecimento da realidade acerca da produção industrial.

Esse conhecimento adquirido permitiu o desenvolvimento incremental de um modelo para o problema, que traduz essa mesma realidade, cumprindo os requisitos e colmatando as necessidades inicialmente identificadas. As alterações realizadas no modelo, ao longo do desenvolvimento do trabalho, revelaram-se pertinentes e necessárias para atingir os objetivos propostos.

O desenvolvimento e implementação do Algoritmo Genético, tendo por base o modelo desenvolvido, apenas foi possível devido ao estudo realizado da tecnologia e análise de diferentes aplicações e vertentes da mesma. O desenvolvimento de diferentes versões e métodos do algoritmo permitiu realizar uma comparação e determinar qual o que melhor se adequava para o problema de planeamento industrial.

A partir dos resultados obtidos, por meio dos testes realizados para comparar diferentes versões, métodos e parametrizações, foi possível tomar decisões importantes na definição da solução definitiva. A nível de parâmetros, conclui-se que, para o problema e implementação em questão, os melhores valores para método de seleção, porção de seleção, porção de elite e probabilidade de mutação são o método do torneio, 0.5, 0.25 e 0.1, respetivamente. Também com base nos resultados obtidos, conclui-se que a melhor versão do algoritmo é o Algoritmo Genético Híbrido, com otimização pelo método de *Tabu Search*. A combinação destas duas técnicas de otimização revelou-se muito útil, na medida em que permitiu obter melhores resultados, a nível da qualidade das soluções geradas, sem aumentar excessivamente o tempo de execução.

Por fim, a validação realizada, através de um caso real, permitiu concluir que o algoritmo, mesmo quando aplicado a um caso real, é capaz de produzir resultados positivos, em comparação com o planeamento real executado pelo fabricante. Melhorias a nível de cumprimento de prazos de entrega, tempo total de *Setup*, quantidade total de *Scrap* e consumo energético total, são possíveis de obter recorrendo ao algoritmo, dependendo das prioridades e objetivos de cada fabricante.

Importante destacar que na aplicação do algoritmo ao caso real, foi possível reduzir o tempo total de produção e aumentar a eficiência de produção para o dobro, na medida em que, o planeamento gerado é capaz de satisfazer o mesmo conjunto de encomendas em metade do tempo do planeamento real executado pela fábrica.

Assim, podemos afirmar que o o balanço final do trabalho é positivo, pois foram cumpridos todos os objetivos inicialmente propostos e os resultados obtidos são promissores, na medida em que, no final, foi possível otimizar a diferentes níveis um planeamento associado a um caso real.

5.2 Trabalho Futuro

O problema de planeamento industrial pode ser tão complexo quanto for desejado e mediante dos objetivos, requisitos e necessidades impostas pelos fabricantes.

Isto leva a que seja necessário um desenvolvimento continuo deste tipo de soluções,

para conseguir acompanhar e colmatar as diferentes necessidades impostas no setor industrial. A aliar a isto, é importante manter um trabalho de investigação e desenvolvimento, na procura de soluções que garantam melhores resultados ou de novas abordagens, sejam novos métodos ou versões alternativas de outros métodos já conhecidos.

O trabalho desenvolvido foi focado na implementação de um Algoritmo Genético para solucionar o planeamento de produção industrial, tendo por base o setor industrial da produção de artigos de plástico, através do método de moldagem por injeção. No futuro é importante continuar o desenvolvimento da solução apresentada, de modo a ser possível aplicar o algoritmo a outras realidades industriais. Estas alterações passam primeiro por acrescentar ao modelo a possibilidade de uma produção possuir várias operações e existir uma sequência de operações a ser cumprida, para cada tipo de produção. Esta é uma característica relevante, que não se aplica ao caso estudado e abordado nesta dissertação, mas é um cenário bastante comum no setor industrial e por isso deve ser abordado em desenvolvimentos futuros. Depois, consoante o setor industrial que for abordado, pode ser necessário adaptar o modelo para satisfazer outras necessidades impostas, mas a adaptação da solução a diferentes cenários é um dos desafios associados ao desenvolvimento deste tipo de ferramenta de planeamento industrial.

Também em perspetiva de trabalho futuro, é importante agregar as diferentes vertentes do planeamento industrial e desenvolver uma solução conjunta e não só focar no planeamento de produção. A possibilidade de otimizar em conjunto o planeamento de produção, manutenção, limpeza, recursos humanos e gestão de armazém, irá trazer beneficio aos fabricantes, visto que todos estas componentes estão interligadas e a má gestão de um deles pode facilmente comprometer as restantes. Por exemplo, um mau planeamento de manutenções pode causar falhas na produção, na medida em que, por exemplo, uma má gestão de manutenções pode levar à interrupção de produções ou ao aumento dos níveis de *scrap* de uma máquina (causado por falta de manutenção), prejudicando assim a produção. O desenvolvimento de uma ferramenta capaz de realizar e otimizar todos estes planeamentos em conjunto, seria a solução ideal para planeamento industrial.

Assim, é notório que este tema de planeamento industrial é bastante amplo e ainda existe muito trabalho a desenvolver e matéria a explorar, acerca do mesmo. Deste modo, mesmo que o trabalho desenvolvido nesta dissertação tenha cumprido todos os objetivos propostos e atingido resultados positivos, é importante continuar a investigar, melhorar e desenvolver a solução implementada.

Bibliografia

[1] A. Abdelghany, K. Abdelghany e F. Azadian. «Airline flight schedule planning under competition». Em: *Computers Operations Research* 87 (2017), pp. 20–39. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2017.05.013. URL: https://www.sciencedirect.com/science/article/pii/S0305054817301259 (ver p. 18).

[2] E. Ahmadi et al. «A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms». Em: *Computers Operations Research* 73 (2016), pp. 56–66. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2016.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0305054816300600 (ver pp. 24–26).

[3] T. Alam et al. «Genetic Algorithm: Reviews, Implementations, and Applications». Em: (2020). DOI: 10.48550/ARXIV.2007.12673. URL: https://arxiv.org/abs/2007.12673 (ver p. 12).

[4] L. Alzubaidi et al. «Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions». Em: *Journal of big Data* 8 (2021), pp. 1–74 (ver p. 19).

[5] «Application of Hybrid Genetic Algorithm Routine in Optimizing Food and Bioengineering Processes». Em: *Foods* 5.4 (2016), pp. 76–. DOI: 10.3390/FOODS5040076 (ver p. 48).

[6] O. H. Babatunde et al. «A genetic algorithm-based feature selection». Em: (2014) (ver p. 19).

[7] B. M. Baker e M. Ayechew. «A genetic algorithm for the vehicle routing problem». Em: *Computers Operations Research* 30.5 (2003), pp. 787–800. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(02)00051-5. URL: https://www.sciencedirect.com/science/article/pii/S0305054802000515 (ver p. 15).

[8] S. Chakraborty e S. Bhowmik. «Job shop scheduling using simulated annealing». Em: *First International Conference on Computation and Communication Advancement*. Vol. 1. 1. 2013, pp. 69–73 (ver p. 27).

[9] M. A. Coletti, E. O. Scott e J. K. Bassett. «Library for Evolutionary Algorithms in Python (LEAP)». Em: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. GECCO '20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 1571–1579. ISBN: 9781450371278. DOI: 10.1145/3377929.3398147. URL: https://doi.org/10.1145/3377929.3398147 (ver p. 13).

[10] H. Cui et al. «Route optimization in township logistics distribution considering customer satisfaction based on adaptive genetic algorithm». Em: *Mathematics and Computers in Simulation* 204 (2023), pp. 28–42. ISSN: 0378-4754. DOI: https://doi.org/10.1016/j.matcom.2022.05.020. URL: https://www.sciencedirect.com/science/article/pii/S0378475422002087 (ver pp. 15–17).

[11] G. Da Col e E. C. Teppan. «Industrial-size job shop scheduling with constraint programming». Em: *Operations Research Perspectives* 9 (2022), p. 100249. ISSN: 2214-7160. DOI: https://doi.org/10.1016/j.orp.2022.100249. URL: https://www.sciencedirect.com/science/article/pii/S2214716022000215 (ver p. 25).

[12] S. Dauzère-Pérès et al. «The flexible job shop scheduling problem: A review». Em: *European Journal of Operational Research* (2023). ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2023.05.017. URL: https://www.sciencedirect.com/science/article/pii/S037722172300382X (ver pp. 22, 24, 25).

[13] P. R. de Oliveira da Costa et al. «A Genetic Algorithm for a Green Vehicle Routing Problem». Em: *Electronic Notes in Discrete Mathematics* 64 (2018). 8th International Network Optimization Conference - INOC 2017, pp. 65–74. ISSN: 1571-0653. DOI: https://doi.org/10.1016/j.endm.2018.01.008. URL: https://www.sciencedirect.com/science/article/pii/S1571065318300088 (ver p. 15).

[14] M. Dell'Amico e M. Trubian. «Applying tabu search to the job-shop scheduling problem». Em: *Annals of Operations research* 41.3 (1993), pp. 231–252 (ver pp. 28, 49).

[15] A. F. Gad. *PyGAD: An Intuitive Genetic Algorithm Python Library*. 2021. DOI: 10.48550/ARXIV.2106.06158. URL: https://arxiv.org/abs/2106.06158 (ver p. 13).

[16] L. Gao et al. «Solving flexible job shop scheduling problem using general particle swarm optimization». Em: *Proceedings of the 36th CIE conference on computers & industrial engineering*. Citeseer. 2006, pp. 3018–3027 (ver p. 27).

[17] F. Garza-Santisteban et al. «A Simulated Annealing Hyper-heuristic for Job Shop Scheduling Problems». Em: *2019 IEEE Congress on Evolutionary Computation (CEC)*. 2019, pp. 57–64. DOI: 10.1109/CEC.2019.8790296 (ver p. 27).

[18] F. Geyik e I. H. Cedimoglu. «The strategies and parameters of tabu search for job-shop scheduling». Em: *Journal of intelligent manufacturing* 15 (2004), pp. 439–448 (ver pp. 28, 49).

[19] B. S. Girish e N. Jawahar. «A particle swarm optimization algorithm for flexible job shop scheduling problem». Em: *2009 IEEE International Conference on Automation Science and Engineering*. 2009, pp. 298–303. DOI: 10.1109/COASE.2009.5234153 (ver p. 27).

[20] G. Gong et al. «A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators». Em: *Journal of Cleaner Production* 174 (2018), pp. 560–576. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2017.10.188. URL: https://www.sciencedirect.com/science/article/pii/S0959652617324952 (ver pp. 24–26).

[21] J. Heinonen e F. Pettersson. «Hybrid ant colony optimization and visibility studies applied to a job-shop scheduling problem». Em: *Applied Mathematics and Computation* 187.2 (2007), pp. 989–998. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2006.09.023. URL: https://www.sciencedirect.com/science/article/pii/S0096300306012355 (ver p. 27).

[22] J. H. Holland. «Genetic Algorithms». Em: *Scientific American* 267.1 (1992), pp. 66–73. ISSN: 00368733, 19467087. URL: http://www.jstor.org/stable/24939139 (ver p. 9).

[23] *Horizon 2020 - EU Funding Program*. URL: https://research-and-innovation.ec.europa.eu/funding/funding-opportunities/funding-programmes-and-open-calls/horizon-2020_en (ver p. 3).

[24] D. L. Hull. «Deconstructing Darwin: Evolutionary theory in context». Em: *Journal of the History of Biology* 38 (2005). ISSN: 1573-0387. DOI: 10.1007/s10739-004-6514-1. URL: https://doi.org/10.1007/s10739-004-6514-1 (ver p. 10).

[25] F. Hussein, N. Kharma e R. Ward. «Genetic algorithms for feature selection and weighting, a review and study». Em: *Proceedings of sixth international conference on document analysis and recognition*. IEEE. 2001, pp. 1240–1244 (ver p. 19).

[26] C. Jacob. «3 - Genetic Algorithms». Em: *Illustrating Evolutionary Computation with Mathematica*. Ed. por C. Jacob. The Morgan Kaufmann Series in Artificial Intelligence. San Diego: Academic Press, 2001, pp. 79–209. ISBN: 978-1-55860-637-1. DOI: https://doi.org/10.1016/B978-155860637-1/50016-3. URL: https://www.sciencedirect.com/science/article/pii/B9781558606371500163 (ver pp. 10, 12).

[27] S. Jaskó et al. «Development of manufacturing execution systems in accordance with Industry 4.0 requirements: A review of standard- and ontology-based methodologies and tools». Em: *Computers in Industry* 123 (2020). ISSN: 0166-3615. DOI: https://doi.org/10.1016/j.compind.2020.103300. URL: https://www.sciencedirect.com/science/article/pii/S0166361520305340 (ver pp. 2, 29).

[28] K. Jebari e M. Madiafi. «Selection methods for genetic algorithms». Em: *International Journal of Emerging Sciences* 3.4 (2013), pp. 333–344 (ver p. 11).

[29] S. Katoch, S. S. Chauhan e V. Kumar. «A review on genetic algorithm: past, present, and future». Em: *Multimedia Tools and Applications* 80 (2021). ISSN: 1573-7721. DOI: 10.1007/s11042-020-10139-6. URL: https://doi.org/10.1007/s11042-020-10139-6 (ver pp. 9–11).

[30] *KITT4SME - platform-enabled KITs of arTificial intelligence FOR an easy uptake by SMEs*. DOI: https://doi.org/10.3030/952119. URL: https://cordis.europa.eu/project/id/952119 (ver p. 3).

[31] M. Kumar et al. «Genetic Algorithm: Review and Application». Em: (2010). DOI: https://dx.doi.org/10.2139/ssrn.3529843. URL: https://ssrn.com/abstract=3529843 (ver pp. 9–12).

[32] R. Leardi. «3 - Genetic Algorithms in Feature Selection». Em: *Genetic Algorithms in Molecular Modeling*. Ed. por J. Devillers. Principles of QSAR and Drug Design. London: Academic Press, 1996, pp. 67–86. ISBN: 978-0-12-213810-2. DOI: https://doi.org/10.1016/B978-012213810-2/50004-9. URL: https://www.sciencedirect.com/science/article/pii/B9780122138102500049 (ver p. 19).

[33] K. C. W. Lim, L.-P. Wong e J. F. Chin. «Hyper-heuristic for flexible job shop scheduling problem with stochastic job arrivals». Em: *Manufacturing Letters* 36 (2023), pp. 5–8. ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2022.12.009. URL: https://www.sciencedirect.com/science/article/pii/S2213846323000044 (ver p. 22).

[34] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (ver p. ii).

[35] S. Loussaief e A. Abdelkrim. «Convolutional neural network hyper-parameters optimization based on genetic algorithms». Em: *International Journal of Advanced Computer Science and Applications* 9.10 (2018) (ver p. 19).

[36] Y. Lu, P. Witherell e A. Jones. «Standard connections for IIoT empowered smart manufacturing». Em: *Manufacturing Letters* 26 (2020). ISSN: 2213-8463. DOI: https://doi.org/10.1016/j.mfglet.2020.08.006. URL: https://www.sciencedirect.com/science/article/pii/S2213846320301449 (ver pp. 1, 29).

[37] S. Luo, L. Zhang e Y. Fan. «Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization». Em: *Journal of Cleaner Production* 234 (2019), pp. 1365–1384. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.06.151. URL: https://www.sciencedirect.com/science/article/pii/S0959652619321110 (ver p. 26).

[38] Y. Matsuo et al. «Deep Learning, Reinforcement Learning, and World Models». Em: *Neural Netw.* 152.C (2022-08), pp. 267–275. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2022.03.037. URL: https://doi.org/10.1016/j.neunet.2022.03.037 (ver p. 19).

[39] Z. Michalewicz e M. Schoenauer. «Evolutionary Algorithms for Constrained Parameter Optimization Problems». Em: *Evolutionary Computation* 4.1 (1996-03), pp. 1–32. ISSN: 1063-6560. DOI: 10.1162/evco.1996.4.1.1. eprint: https://direct.mit.edu/evco/article-pdf/4/1/1/1492890/evco.1996.4.1.1.pdf. URL: https://doi.org/10.1162/evco.1996.4.1.1 (ver p. 10).

[40] H. Mokhtari e A. Hasani. «An energy-efficient multi-objective optimization for flexible job-shop scheduling problem». Em: *Computers Chemical Engineering* 104 (2017), pp. 339–352. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2017.05.004. URL: https://www.sciencedirect.com/science/article/pii/S0098135417302030X (ver pp. 24–26).

[41] *Muvu Technologies*. URL: https://muvu.tech/ (ver pp. 1, 2).

[42] E. Y. Nakagawa et al. «Industry 4.0 reference architectures: State of the art and future trends». Em: *Computers Industrial Engineering* 156 (2021). ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2021.107241. URL: https://www.sciencedirect.com/science/article/pii/S0360835221001455 (ver p. 1).

[43] M. Nouiri et al. «An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem». Em: *Journal of Intelligent Manufacturing* 29.3 (2018-03), pp. 603–615. ISSN: 1572-8145. DOI: 10.1007/s10845-015-1039-3. URL: https://doi.org/10.1007/s10845-015-1039-3 (ver p. 27).

[44] C. S. Perone. «Pyevolve: A Python Open-Source Framework for Genetic Algorithms». Em: *SIGEVOlution* 4.1 (2009-11), pp. 12–20. DOI: 10.1145/1656395.1656397. URL: https://doi.org/10.1145/1656395.1656397 (ver p. 13).

[45] A. Sahli et al. «An effective and robust genetic algorithm for urban freight transport scheduling using passenger rail network». Em: *Computers Industrial Engineering* 173 (2022), p. 108645. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2022.108645. URL: https://www.sciencedirect.com/science/article/pii/S0360835222006337 (ver pp. 15, 16).

[46] A. Segerstedt. «A simple heuristic for vehicle routing – A variant of Clarke and Wright's saving method». Em: *International Journal of Production Economics* 157 (2014). The International Society for Inventory Research, 2012, pp. 74–79. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2013.09.017. URL: https://www.sciencedirect.com/science/article/pii/S0925527313004222 (ver p. 15).

[47] «Selected Soft Computing Algorithms for Job Shop Problem (JSP)». Em: *International Journal of Science and Engineering Applications* 10.9 (2021), pp. 125–131. DOI: 10.7753/IJSEA1009.1003 (ver pp. 21, 22).

[48] H. D. Sherali, E. K. Bish e X. Zhu. «Airline fleet assignment concepts, models, and algorithms». Em: *European Journal of Operational Research* 172.1 (2006), pp. 1–30. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2005.01.056. URL: https://www.sciencedirect.com/science/article/pii/S0377221705002109 (ver p. 18).

[49] B. Sigl, M. Golub e V. Mornar. «Solving timetable scheduling problem using genetic algorithms». Em: *Proceedings of the 25th International Conference on Information Technology Interfaces, 2003. ITI 2003.* 2003, pp. 519–524. DOI: 10.1109/ITI.2003.1225396 (ver pp. 17, 18).

[50] B. Silva et al. «Enhance the Injection Molding Quality Prediction with Artificial Intelligence to Reach Zero-Defect Manufacturing». Em: *Processes* 11.1 (2023). ISSN: 2227-9717. DOI: 10.3390/pr11010062. URL: https://www.mdpi.com/2227-9717/11/1/62 (ver p. 2).

[51] H. Singh. «Big data, industry 4.0 and cyber-physical systems integration: A smart industry context». Em: *Materials Today: Proceedings* 46 (2021). 2nd International Conference on Manufacturing Material Science and Engineering. ISSN: 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2020.07.170. URL: https://www.sciencedirect.com/science/article/pii/S2214785320352639 (ver p. 1).

[52] Y. Sun et al. «Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification». Em: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3840–3854. DOI: 10.1109/TCYB.2020.2983860 (ver p. 19).

[53] P. Victer Paul et al. «Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems». Em: *Applied Soft Computing* 32 (2015), pp. 383–402. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.03.038. URL: https://www.sciencedirect.com/science/article/pii/S1568494615001969 (ver p. 51).

[54] A. S. Wicaksono e A. A. Supianto. «Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction». Em: *International Journal of Advanced Computer Science and Applications* 9.12 (2018) (ver p. 19).

[55] N. Wiriyasermkul, V. Boobjing e P. Chanvarasuth. «A Meiosis Genetic Algorithm». Em: *2010 Seventh International Conference on Information Technology: New Generations*. 2010, pp. 285–289. DOI: 10.1109/ITNG.2010.152 (ver p. 10).

[56] J. Wu, G. Wu, J. Wang et al. «Flexible job-shop scheduling problem based on hybrid ACO algorithm». Em: *International Journal of Simulation Modelling* 16.3 (2017), pp. 497–505 (ver p. 27).

[57] X. Wu e Y. Sun. «A green scheduling algorithm for flexible job shop with energy-saving measures». Em: *Journal of Cleaner Production* 172 (2018), pp. 3249–3264. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2017.10.342. URL: https://www.sciencedirect.com/science/article/pii/S0959652617326483 (ver p. 26).

[58] S. Yang. «Using Attention Mechanism to Solve Job Shop Scheduling Problem». Em: *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*. 2022, pp. 59–62. DOI: 10.1109/ICCECE54139.2022.9712705 (ver p. 22).

[59] M. Yoshikawa et al. «A constraint-based high school scheduling system». Em: *IEEE Expert* 11.1 (1996), pp. 63–72. DOI: 10.1109/64.482960 (ver p. 17).

[60] S. R. Young et al. «Optimizing deep learning hyper-parameters through an evolutionary algorithm». Em: *Proceedings of the workshop on machine learning in high-performance computing environments*. 2015, pp. 1–5 (ver p. 19).

[61] J. Zhou e Z. Hua. «A correlation guided genetic algorithm and its application to feature selection». Em: *Applied Soft Computing* 123 (2022), p. 108964. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2022.108964. URL: https://www.sciencedirect.com/science/article/pii/S1568494622003052 (ver p. 19).

[62] R. Zhou, H. P. Lee e A. Y. Nee. «Applying Ant Colony Optimisation (ACO) algorithm to dynamic job shop scheduling problems». Em: *International Journal of Manufacturing Research* 3.3 (2008), pp. 301–320 (ver p. 27).