NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
ENGENHARIA MECÂNICA E INDUSTRIAL

ANA RAQUEL MARQUES MADUREIRA

Licenciada em Ciências de Engenharia e Gestão Industrial

CALENDARIZAÇÃO ROBUSTA DE VEÍCULOS AUTÓNOMOS ATRAVÉS DE PROGRAMAÇÃO LINEAR INTEIRA MISTA

APLICAÇÃO INDUSTRIAL AO PROJETO AGILE

MESTRADO INTEGRADO EM ENGENHARIA E GESTÃO INDUSTRIAL

Universidade NOVA de Lisboa
Abril, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTAMENTO DE
ENGENHARIA MECÂNICA E INDUSTRIAL

CALENDARIZAÇÃO ROBUSTA DE VEÍCULOS AUTÓNOMOS ATRAVÉS DE PROGRAMAÇÃO LINEAR INTEIRA MISTA

APLICAÇÃO INDUSTRIAL AO PROJETO AGILE

ANA RAQUEL MARQUES MADUREIRA

Licenciada em Ciências de Engenharia e Gestão Industrial

Orientadora: Professora Doutora Alexandra Maria Baptista Ramos Tenera
Professora Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Coorientador: Professor Doutor Ricardo Pinto Moura
Professor Auxiliar Convidado, Instituto Superior de Economia e Gestão da Universidade de Lisboa

Júri

Presidente: Professora Doutora Ana Sofia Matos
Professora Associada, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Arguente: Professor Doutor Manuel Valdemar Cabral Vieira
Professor Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Vogal: Engenheiro João Pedro Baptista Mendes
Gestor do Departamento de Automação e Robótica, Imeguisa Portugal

MESTRADO INTEGRADO EM ENGENHARIA E GESTÃO INDUSTRIAL

Universidade NOVA de Lisboa
Abril, 2024

Calendarização Robusta de Veículos Autónomos através de Programação Linear Inteira Mista
Aplicação Industrial ao Projeto AGiLE

Este documento foi gerado com o processador (pdf/Xe/Lua)LaTeX e o modelo NOVAthesis (v6.10.18) (Lourenço, 2021).

À minha família e amigos.

Agradecimentos

Primeiramente, gostaria de expressar minha gratidão á Professora Alexandra Tenera e ao Professor Ricardo Moura pela sua orientação e apoio inestimáveis ao longo deste processo. Obrigada também à Mina Norouzirad pela dedicação do seu tempo e conhecimento, por me ouvir e orientar, sempre que precisei.

Um especial agradecimento ao João Mendes e à Marta Filipe pelo apoio e disponibilidade incondicionais, pela motivação e confiança nos momentos em que mais precisei. A todos na Imeguisa Portugal, um sincero obrigado pela receção tão calorosa e por me fazerem sentir em casa desde o momento em que comecei este trabalho.

Agradeço à minha família, pois sem o apoio deles, esta conquista não teria sido possível. Obrigada aos meus pais Fernanda e Luís pelo constante apoio, encorajamento e compreensão. Devo-lhes tudo o que conquistei até agora. Obrigada ao meu pai Rui por sua confiança em mim, por estar sempre disposto a ouvir e por me encorajar a perseverar e nunca desistir sem lutar. Obrigada aos meus irmãos Filipa, Luísa e Miguel por me aturarem quando estava de mau humor, mas principalmente por serem as pessoas que trazem alegria à minha vida e me inspiram a ser um exemplo todos os dias. Obrigada aos meus avós Ilda, Antonieta, Laura, Heitor e António por fazerem grande parte da pessoa que sou hoje.

Obrigada à Carolina por acreditar em mim e por me fazer ver que posso alcançar a vida com que sempre sonhei. Obrigada por seres um exemplo notável de determinação e dedicação à busca pelo que nos faz mais felizes.

Obrigada à Liliana, que tem sido a minha rocha desde os meus 10 anos, que me apoiou nos momentos mais difíceis, mas também me mostrou que a vida é mais rica quando compartilhamos risos, brincadeiras e tempo com aqueles que mais amamos.

Obrigada à Adame pela sua amizade desde que entrei na faculdade. Foi a primeira pessoa com quem me identifiquei em Engenharia Física, com quem partilhei tantos momentos incríveis e me faz sentir ouvida e compreendida. Obrigada pela tua companhia e amizade inabalável.

Obrigada à Diana, Vasco, Marta e Patrícia por terem feito a transição para MIEGI muito mais fácil. Estiveram sempre ao meu lado e não só tornaram tudo mais divertido, como também deram um novo significado à minha experiência na faculdade.

Por fim, mas nunca menos importante, quero agradecer à minha madrinha Leonor, Moura, Audrey, Capela, Newton, Patrícia, à 2.0 e aos meus afilhados Eduardo, Teresa, Joana e Mafalda por fazerem toda a diferença na minha vida. Agradeço todos os dias por vos ter conhecido, por me fazerem ficar e por me apoiarem incondicionalmente. Devo muito à vossa presença na minha vida.

A todos vocês, Obrigada.

”

«*No one can see the future. I can see options and trajectories. Time is like a river that splits into creeks or pools into lakes or careens down waterfalls. I have the map, and I steer the ship.*»

— Garnet (Gem), Steven Universe, Episódio *Future Vision*

Resumo

A presente dissertação introduz a calendarização de veículos autónomos no contexto do projeto AGiLE, focado no abastecimento descentralizado de uma linha de montagem através de *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR)s e periféricos sensorizados. Com este projeto pretende-se obter um abastecimento completamente reativo às necessidades da linha, cujas vantagens incluem a capacidade de manter um nível de serviço constante em linhas de produção ou montagem. No entanto, está também associado a desvantagens, como a ocorrência de falhas de comunicação entre robôs ou exclusão de robôs prestes a terminar tarefas e cuja inclusão iria diminuir o tempo de execução da tarefa a realizar.

Neste contexto, pretende-se desenvolver um modelo de Calendarização Robusta com recurso à Programação Linear Inteira Mista, de forma a construir um planeamento preditivo e robusto a eventos imprevistos, como avarias dos robôs e localização dos mesmos a cada momento. Pressupõe-se a utilização de expressões lineares para alocar n tarefas a um conjunto de m robôs, de forma a minimizar tempos de espera e duração prevista das tarefas a alocar.

Foram testadas várias possibilidades de calendarização, com diversos conjuntos de tarefas e robôs. Conseguiu-se a alocação de 2 tarefas a múltiplos AMRs, o que demonstra o grande potencial do método desenvolvido. No entanto, os resultados obtidos até o momento, para mais de 2 tarefas, não apresentam conclusões definitivas sobre a viabilidade da aplicação dessa metodologia.

Os desafios encontrados incluem a ocorrência de problemas numéricos que infringem a integralidade das variáveis de decisão, a linearização de expressões através do método *Big-M* que leva a alocação de tarefas com sobreposições no mesmo robô e ainda o controlo dos tempos de início e fim de uma tarefa, que torna difícil a adaptação do método *Big-M* a este problema. A estes junta-se a dificuldade de interpretação do *output* do *solver*, que se limita aos valores das variáveis de decisão e de representações lineares da restrições, após a otimização.

A superação dos desafios técnicos e a compreensão das limitações da metodologia são passos importantes para o desenvolvimento de uma solução robusta e aplicável a cenários reais, num futuro próximo.

Palavras-chave: Veículo Autónomo Móvel, Veículo Guiado Automático, Calendarização Robusta, Programação Linear Inteira Mista, Simulação de Eventos Discretos

Abstract

This dissertation introduces scheduling of autonomous vehicles in the context of the AGiLE project, focused on decentralized supply to an assembly line through *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR)s and sensorized peripherals. With this project we intend to achieve a supply system that reacts completely to the needs of the assembly line, with advantages including the ability to maintain a constant service level in production/assembly lines. However, it is also associated with disadvantages, such as communication failures between robots or the exclusion of robots about to complete tasks, which could reduce task completion time.

In this context, the goal is to develop a Robust Scheduling methodology using Mixed-Integer Linear Programming to create a schedule resilient to unforeseen events, such as robot breakdowns and their location during task execution. Linear expressions are assumed to allocate 'n' tasks to a set of 'm' robots, minimizing waiting times and the expected duration of the allocated tasks.

Various possibilities of scheduling have been tested, with various sets of tasks and robots. The allocation of 2 tasks to multiple AMRs was achieved, which demonstrates the great potential of the method developed. However, the results obtained so far, for more than 2 tasks, do not present definitive conclusions on the feasibility of applying this methodology.

Challenges encountered include numerical problems that compromise the integrality of decision variables, linearizing expressions using the 'Big-M' method, leading to task overlaps on the same robot, and the control of task start and end times, making it difficult to adapt the "Big-M" method to this problem. Added to these challenges is the difficulty in interpreting the solver's output, which is limited to decision variable values and linear representations of constraints, after optimization.

Overcoming technical challenges and understanding the limitations of the methodology are important steps in developing a robust solution applicable to real-world scenarios in the near future.

Keywords: Autonomous Mobile Robots, Automatic Guided Vehicle, Robust Scheduling, Mixed Integer Linear Programming, Discreet Event Simulation

Índice

Índice de Figuras

Índice de Tabelas

SIGLAS

ACR-LH	Aro da Cava da Roda do lado esquerdo
ACR-RH	Aro da Cava da Roda do lado direito
AD	Teste de *Anderson-Darling*
AE	Volkswagen Autoeuropa, Lda.
AGV	*Automated Guided Vehicle* ou Veículos Automáticos Guiados
AIC	*Akaike Information Criterion* ou Critério de Informação de Akaike
AMR	*Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos
BIC	*Bayesian or Schwarz Information Criterion* ou Critério de Informação Bayesiano ou de Schwarz
DES	*Discreet Event Simulation* ou Simulação de Eventos Discretos
DSR	*Design Science Research*
FIFO	*First In First Out*
FJSP	*Flexible Job Shop Problem* ou Problema de *Job Shop* Flexível
GMM	*Gaussian Mixture Model* ou Modelo de Mistura Gaussiana
GRB	Gurobi Optimizer
KDE	*Kernel Density Estimation* ou Estimativa de Densidade de Kernel
KPI	*Key Performance Indicator*
KS	Teste de *Kolmogorov-Smirnov*
MILP	*Mixed Integer Linear Programming* ou Programação Linear Inteira Mista
MM	Módulos do Motor
PFSP	*Permutation Flow Shop Problem* ou Problema de Permutação de *Flow Shop*
PMP	*Parallel Machine Problem* ou Problema de Máquinas Paralelas
POF	*Point of Fit*
RLT	*Reformulation-Linearization Technique* ou Técnica de Reformulação-Linearização
RS	*Robust Scheduling* ou Calendarização Robusta

SUMA Supermercado de manufatura

TF Tampas e Filtros

UPMP *Unrelated Parallel Machines Problem* ou Problema de Máquinas Paralelas não relacionadas

WIP *Work in Progress*

Capítulo 1

Introdução

1.1 Contextualização do Problema

A Quarta Revolução Industrial impulsionou a adoção de tecnologias da *Internet-of-Things*, permitindo o aumento da competitividade de uma empresa e da sua relevância no mercado e traçando o caminho para a globalização da tecnologia da Indústria 4.0 (Mahmood et al., 2021). Estes combinam a engenharia de produção, sistemas de informação e comunicação, e análise de *Big Data*, de forma a transformar a produção em massa na produção de pequenos lotes e uma ampla variedade de produtos. Para o efeito, é necessário aumentar a eficácia e eficiência das operações intralogísticas com a utilização de agentes como os *Automated Guided Vehicle* ou Veículos Automáticos Guiados (AGV) e *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR), veículos autónomos capazes de se adaptarem a novos e dinâmicos ambientes fabris. Esta pressão exige também o desenvolvimento de novas tecnologias de planeamento de rotas e calendarização de tarefas destes mesmos robôs (Fragapane et al., 2022).

A calendarização de recursos como os AGVs e AMRs requer a integração de variáveis (como obstáculos dinâmicos) e da ocorrência de eventos inesperados, como por exemplo avarias e incerteza da duração de tarefas a executar (Levorato et al., 2023). *Robust Scheduling* ou Calendarização Robusta (RS) é introduzida como uma possível solução para o planeamento das operações de transporte e abastecimento de componentes a linhas de produção e montagem através de veículos autónomos. A *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP) é reconhecida como uma ferramenta de RS flexível e adaptável a diversos problemas, tipos de atividades, recursos a utilizar e medidas de desempenho. Esta pressupõe a utilização de expressões lineares para representar as medidas de desempenho a otimizar, bem como os pressupostos do problema na forma de equações e inequações lineares (Hillier & Lieberman, 2015).

A formulação do problema de calendarização em questão como um problema MILP pressupõe a utilização de um método de resolução do problema, de forma a obter uma solução ótima. Dada a dimensão deste tipo de problemas, é comum a aplicação de um *solver*, como o

Gurobi Optimizer (GRB), para encontrar a alocação ótima de tarefas aos recursos disponíveis, respeitando as restrições lineares impostas (Gurobi Optimization, 2023a).

O modelo de Calendarização Robusta será desenvolvido e testado no âmbito do Projeto AGiLE, motivado pelo atual abastecimento puramente reativo por parte de uma frota de robôs, de forma a satisfazer as necessidades de componentes de uma linha de montagem.

1.2 Projeto AGiLE

A Imeguisa Portugal - Indústrias Metálicas Reunidas, S.A., membro do Grupo Imeguisa (sediado em Espanha), conta com uma vasta experiência em equipamentos de Intralogística, nomeadamente comboios logísticos, *racks*, contentores e veículos automatizados. Oferece soluções de acondicionamento e transporte de componentes, armazenamento e mobiliário industrial ajustadas ás necessidades do cliente, sendo esta personalização e serviço de qualidade a força motriz por trás do crescimento que a empresa sofreu nos últimos anos.

O projeto AGiLE, idealizado pela Imeguisa Portugal, e em parceria com a Volkswagen Autoeuropa, Lda. (AE) e o Instituto Superior Técnico, visa automatizar os processos intralogísticos em linhas de produção e de montagem. Foca-se na automatização do transporte entre o armazém e o local de montagem dos componentes, através da comunicação entre os agentes de transporte e periféricos, garantindo o fornecimento eficaz de componentes e melhorando o uso dos recursos disponíveis. O sistema é composto por três veículos autónomos, periféricos sensorizados e uma entidade supervisora, todos conectados numa arquitetura em rede. Esta abordagem procura escalabilidade e flexibilidade na implementação e funcionamento do sistema, onde todos os AMRs têm a capacidade de navegação autónoma e comunicação com os periféricos e a entidade supervisora, independentemente do ambiente de produção em que se encontram (Imeguisa Portugal, 2021).

Na fase de desenvolvimento do projeto, Almeida (2022) estudou o desempenho do abastecimento da linha de montagem da AE usando o sistema original do Projeto AGiLE. Para o efeito, foi utilizada a *Discreet Event Simulation* ou Simulação de Eventos Discretos (DES), através do *software Flexsim*, para observar o comportamento esperado do sistema e determinar o número ideal de robôs que satisfaz as necessidades atuais da linha de montagem.

1.2.1 Motivação e Objetivos

Como mencionado, o AGiLE pressupõe a conexão estável entre robôs e com os periféricos de transporte. Mais ainda, a alocação de tarefas aos robôs baseia-se no critério "AMR livre mais próximo" do periférico a transportar, através de um "leilão" entre todos os robôs livres da frota, pelo que não tem em conta outros robôs ocupados, mas prestes a terminar a tarefa corrente. Estas considerações motivam o desenvolvimento de uma metodologia de calendarização e alocação de tarefas aos AMRs integrantes do sistema, de forma a eliminar a necessidade de comunicação entre todos os robôs aquando da alocação de uma tarefa a um robô, bem como a utilização das tarefas anteriores como indicação da posição de um robô durante o planeamento e, como consequência, da duração das tarefas seguintes.

O objetivo do presente estudo é o desenvolvimento de um modelo de Calendarização Robusta através de Programação Linear Inteira Mista, a implementar no contexto do Projeto AGiLE.

1.3 Metodologia de Estudo

A presente dissertação procura desenvolver um modelo de RS para o abastecimento da linha de montagem final da AE através de veículos autónomos. É imperativo escolher a metodologia que melhor se adequa, de forma a construir uma nova solução de calendarização dos AMRs mais eficaz e eficiente. Escolheu-se assim a metodologia *Design Science Research* (DSR), descrita na Figura 1.1.

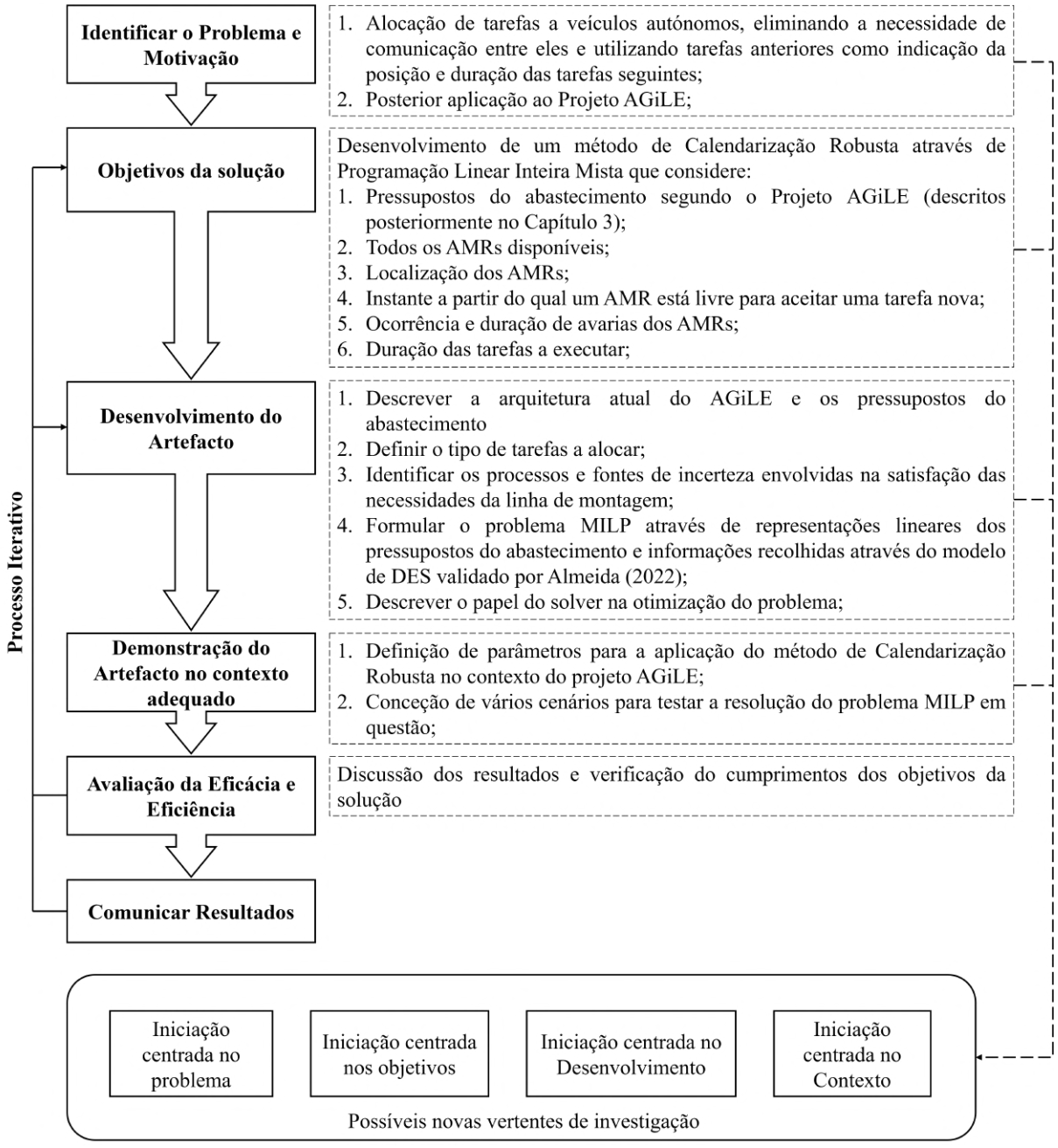


Figura 1.1: Metodologia de *Design Science Research* aplicada ao desenvolvimento do método de Calendarização Robusta. Adaptado de (Vom Brocke et al., 2020).

A DSR é uma metodologia introduzida pela primeira vez em 1969 por Simon (1969) na obra "The Sciences of the Artificial", onde a expressão "Science of Design" foi cunhada pelo autor. Na sua obra, Simon debruça-se sobre os processos que levam à criação e produção do que é artificial, ou seja, uma invenção/criação do ser humano. Abrange vários campos, desde a engenharia à arquitetura. No entanto, é na área dos Sistemas de Informação que se destaca como a metodologia de eleição entre académicos e organizações. Esta metodologia tem como objetivo operacionalizar a solução do problema de alocação de tarefas, garantindo também a geração de novo conhecimento e a posterior generalização da solução a novas vertentes de investigação centradas na motivação do presente estudo, no método de calendarização proposto, no desenvolvimento do problema MILP ou ainda noutro contexto paralelo ao Projeto AGiLE.

A natureza do problema, que pressupõem o desenvolvimento de uma metodologia de calendarização de veículos autónomos com base na implementação atual do Projeto AGiLE, necessita da componente iterativa e de constante reavaliação dos resultados de cada iteração, de forma a perceber se os resultados demonstram que os problemas identificados foram resolvidos. Em cada iteração, o *design* da solução altera-se com o conhecimento adquirido e com a incapacidade de atingir determinados objetivos do planeamento pretendido.

1.4 Estrutura da Dissertação

A metodologia DSR proporciona a estruturação da presente dissertação através das suas fases e processos de iteração inerentes.

O Capítulo 1 salienta o contexto em que se insere o estudo e o projeto que questionou a integração da calendarização de veículos autónomos como possível solução para a alocação de tarefas aos robôs. Introduz também a metodologia DSR e, consequentemente, a estrutura da dissertação.

O Capítulo 2 reflete sobre as características dos veículos utilizados, os desafios na sua implementação e no papel da Programação Linear Inteira Mista como metodologia de Calendarização Robusta.

O Capítulo 3 dá a conhecer o projeto AGiLE com maior pormenor, ao explicar a sua estrutura, pressupostos e características necessárias para compreender a necessidade de uma nova forma de alocação de tarefas. Introduz algumas propostas de alteração do modelo de simulação construído previamente por Almeida (2022).

O Capítulo 4 condensa as fases de "Identificar o Problema e Motivação", "Objetivos da solução" e "Desenvolvimento do Artefacto" da DSR, esta última como ponto de partida para as restantes iterações da metodologia. Introduz o problema de Máquinas Paralelas não relacionadas, cuja formulação matemática através de MILP leva a uma abordagem de calendarização preditiva-adaptativa de n tarefas a m robôs. O *solver* Gurobi Optimizer é utilizado para resolver o modelo construído em *Python* e encontrar uma solução ótima para o planeamento.

O Capítulo 5 compila todas as iterações da metodologia proposta (agregando as fases de "Demonstração do Artefacto no contexto adequado" e "Avaliação da Eficácia e Eficiência"), ao utilizar vários cenários hipotéticos para perceber, em cada ciclo, qual a influência das alterações nos vários planeamentos gerados.

Por fim, no Capítulo 6 é feito o resumo dos resultados alcançados durante a implementação da metodologia ao Projeto AGiLE e identificação de possíveis trabalhos futuros como novas vertentes de investigação.

Capítulo 2

Calendarização Robusta de Veículos Autónomos

Em plena quarta revolução industrial, a implementação de tecnologias da *Internet of Things* (IoT), realidade aumentada e sensores inteligentes em ambientes produtivos é uma força motriz para aumento da competitividade de uma empresa e da sua relevância no mercado. Nos últimos anos tem-se registado um grande avanço das tecnologias computacionais que permitem a recolha e análise de uma grande quantidade de dados, abrindo caminho para a globalização das tecnologias da Indústria 4.0 (Mahmood et al., 2021; Stączek et al., 2021). Estas combinam a engenharia de produção, sistemas de informação e comunicação e ainda a análise de Big Data para atualizar a industria atual: transformar a produção em massa na produção de pequenos lotes e de grande variedade de produtos (Fragapane et al., 2022; Gao et al., 2022). A preferência do mercado pela grande variedade e personalização de produtos implica maior capacidade de resposta das empresas. Para manter a produtividade, as indústrias devem investir em novas linhas de produção ou potenciar as linhas existentes, diminuindo o custo de oportunidade através da melhoria das operações de Intralogística (Afonso et al., 2021; Fragapane et al., 2022).

A Intralogística ou Logística Interna é um ramo da logística focado nas operações dentro de uma instalação. Esta inclui operações de armazenagem e fornecimento de linhas ou unidades de produção, nas quais é focado o presente estudo. Hoje em dia, muitas destas operações ainda são realizadas utilizando mão-de-obra humana e meios de transporte de carga como empilhadores, que constituem um maior custo operacional, de qualidade e de segurança no trabalho, para além de constituírem uma utilização de recursos ineficiente e fonte de desperdícios de vários tipos (Afonso et al., 2021; Mahmood et al., 2021). É nesta perspetiva que se introduzem os *Automated Guided Vehicle* ou Veículos Automáticos Guiados (AGV) e os *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR). Existem ainda outras denominações, com os Veículos Automáticos Inteligentes (Afonso et al., 2021).

Os AGVs e AMRs têm ganho popularidade pelo investimento inicial relativamente reduzido e pelo aumento da eficiência e diminuição do tempo de resposta que conseguem promover nas operações de intralogística de uma fábrica. Quanto maior e mais complexo é o *layout* de

uma instalação, maior será o impacto que a implementação destes sistemas têm no aumento da capacidade produtiva e de resposta ao mercado (Liu et al., 2022). Esta exigência de eficácia e eficiência em ambiente fabril impõe uma enorme pressão sobre o desenvolvimento de novas soluções. O aparecimento de novas tecnologias, como algoritmos para contornar obstáculos, de calendarização e planeamento de rotas, bem como de novos indicadores de desempenho, são alguns exemplos de inovação a explorar no presente capítulo.

Posto isto, pretende-se responder ás seguintes questões de investigação:

Q1: Quais os principais tipos de sistema de navegação e controlo de AGVs and AMRs?

Q2: Quais os desafios encontrados durante a implementação destes sistemas?

Q3: Como é avaliada a performance destes sistemas em ambiente fabril?

Q4: Como é que a Programação Linear Inteira Mista pode ser usada para a formulação de problemas de Calendarização Robusta?

2.1 Análise Bibliométrica e Revisão Bibliográfica

A presente revisão de literatura tem como objetivo a compilação dos principais desafios encontrados na implementação e monitorização de frotas de AGVs/AMRs, enquanto frotas já operacionais ou simuladas. Pretende-se também perceber como a *Robust Scheduling* ou Calendarização Robusta (RS) pode ser utilizada na gestão de operação em diversas áreas e tipos de problemas, utilizando *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP) para a obtenção de uma solução robusta.

2.1.1 Metodologia: Revisão Sistemática

Foram selecionadas publicações de acordo com a metodologia de Revisão Sistemática. Esta pretende identificar as questões de investigação mais pertinentes e suportar a futura investigação no que são as práticas correntes na área em estudo (Torres-Carrion et al., 2018). O processo pode ser dividido em 3 fases: planeamento, revisão e redação. Nas fases de planeamento e revisão, as publicações foram selecionadas, filtradas e analisadas segundo a metodologia descrita no fluxograma da Figura 2.1.

Numa primeira revisão sobre AGVs e AMRs enquanto agentes intralogísticos, pretendeu-se explorar várias editoras, pelo que foi usada a base de indexação *Scopus*. Para as restantes *strings* de *keywords* utilizadas, a pesquisa no *Scopus* mostrou-se insuficiente, dada a especificidade do tema, quando comparada, por exemplo, com o motor de busca *Science Direct*. A título de exemplo, quando pesquisada a *string* (*"robust scheduling"AND "production"AND "machines"*), obtêm-se 42 resultados no *Scopus*, ao passo que no motor de busca *Science Direct* obtêm-se 519 resultados. Assim, para as *strings* seguintes foi apenas utilizado o motor de busca *Science Direct*.

Todas as publicações (compiladas na Tabela 2.1) foram filtradas segundo 3 condições. A primeira considerou apenas artigos empíricos e de revisão. Dado que a última revisão sobre a implementação de AGV / AMR foi publicada em 2022 e, após a análise das 3 revisões encontradas, ter-se verificado que as 3 compilavam informação de grande relevância para o presente estudo,

foram todas analisadas. A revisão de literatura de Oyekanlu et al. (2020) é a mais completa de todas as revisões encontradas nesta área, portanto foi também analisada e adicionada ao conjunto de publicações n_1, na Figura 2.1. A segunda condição restringe as publicações à língua portuguesa e inglesa. Por fim, a terceira condição limita as publicações aos últimos 3 anos (de 2021 a 2023), dada a relevância das revisões encontradas neste período.

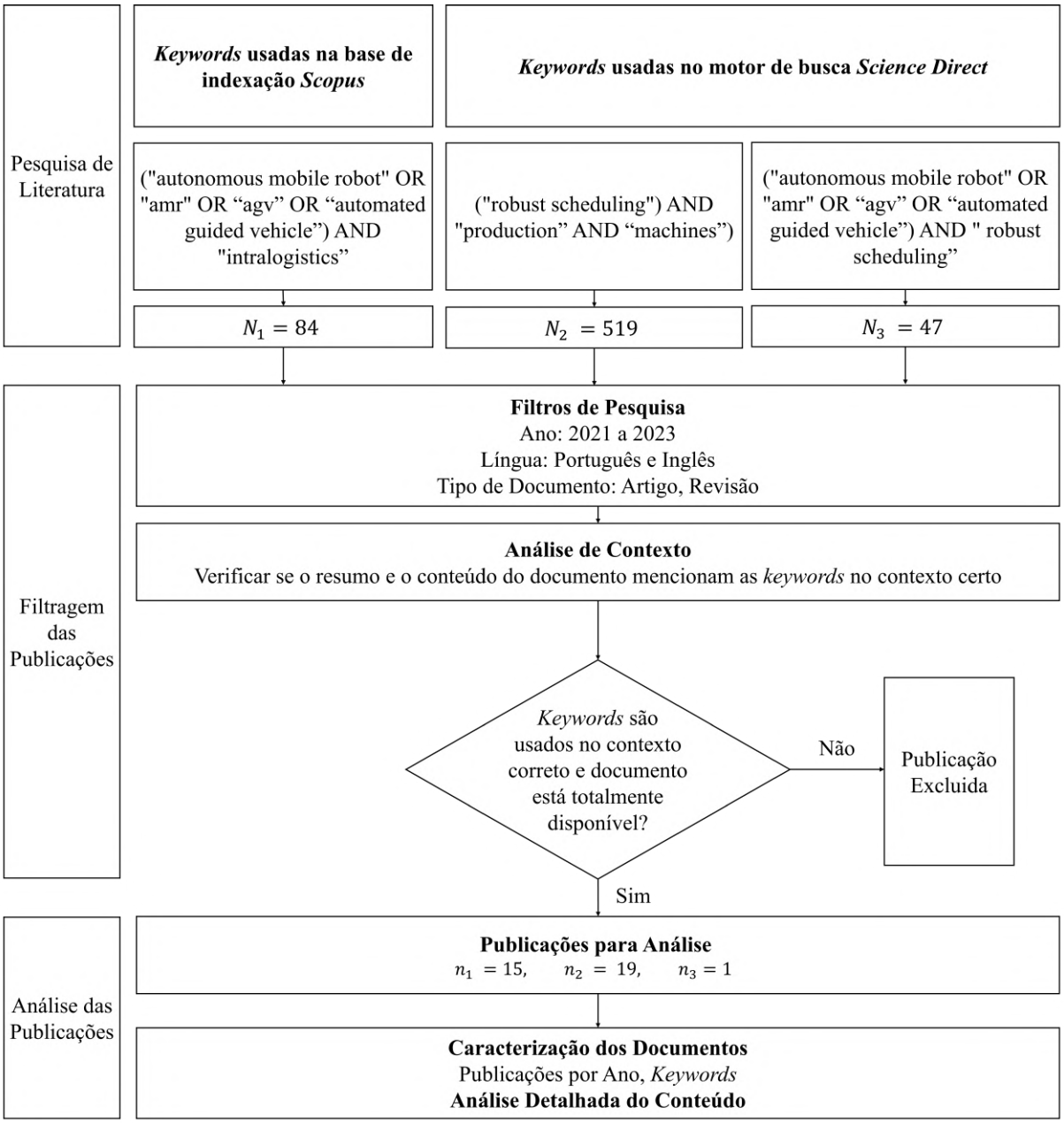


Figura 2.1: Metodologia de revisão de literatura sistemática: fases de planeamento e revisão das publicações.

Tabela 2.1: Características das publicações analisadas. A coluna **Cit** indica o número de citações da publicação. A coluna **EC** indica se a publicação inclui Estudo de Caso real. A abreviatura **N/A** significa "Não Aplicável".

Referência	Cit	Título	EC	Métodos
AGV eAMR como Agentes Intralogísticos (publicações n_1)				
Reis et al., 2023	17	Automated guided vehicles position control: a systematic literature review		N/A
Sperling et al., 2023	0	Classified AGV Material Flow and Layout Data Set for Multidisciplinary Investigation		N/A
Sierra-García et al., 2023	4	Development and Experimental Validation of Control Algorithm for Person-Following Autonomous Robots		
Bekishev et al., 2023	1	FMEA Model in Risk Analysis for the Implementation of AGV/AMR Robotic Technologies into the Internal Supply System of Enterprises	✓	FMEA Model
Kopp et al., 2023	1	Perspectives of managers and workers on the implementation of automated-guided vehicles (AGVs)—a quantitative survey		
Chen et al., 2023	0	Smart navigation via strategic communications in a mixed autonomous paradigm		
Liu et al., 2022	3	A new knowledge-guided multi-objective optimisation for the multi-AGV dispatching problem in dynamic production environments	✓	Knowledge guided multi-objective Estimation of Distribution Algorithm with delivery Satisfaction evaluation
Gao et al., 2022	8	Heterogeneous Multitype Fleet Green Vehicle Path Planning of Automated Guided Vehicle with Time Windows in Flexible Manufacturing System	✓	Hybrid Genetic Algorithm with Large Neighborhood Search (GA-LNS)
Fragapane et al., 2022	117	Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics		Analytical model for a throughput analysis
Stączek et al., 2021	26	A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMRs) Operating Environment—A Case Study	✓	Discreet Event Simulation
Reith et al., 2021	1	Conflict-minimal routing for free-ranging transportation vehicles in in-house logistics based on an a-priori lane design	✓	Lane-based approach for routing of free ranging vehicles
Fragapane et al., 2021	146	Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda		N/A

Continua na página seguinte

Referência	Cit	Título	EC	Métodos
Mahmood et al., 2021	2	Production Intralogistics Automation Based on 3D Simulation Analysis	✓	Discreet Event Simulation
Afonso et al., 2021	8	Simulation pulled by the need to reduce wastes and human effort in an intralogistics project	✓	Discreet Event Simulation
Oyekanlu et al., 2020	131	A Review of Recent Advances in Automated Guided Vehicle Technologies: Integration Challenges and Research Areas for 5G-Based Smart Manufacturing Applications		N/A
Calendarização Robusta (publicações $n_2 + n_3$)				
Cohen et al., 2023	3	An adaptive robust optimization model for parallel machine scheduling		MILP Problem + Two-stage approximation heuristic
Levorato et al., 2023	0	Robust permutation flow shop total weighted completion time problem: Solution and application to the oil and gas industry	✓	Hybrid MILP + Branch-and-Bound solution method using CPLEX Solver
Peng et al., 2023	2	Critical chain based Proactive-Reactive scheduling for Resource-Constrained project scheduling under uncertainty		Critical Chain Method
Ghaleb e Taghipour, 2023	0	Dynamic shop-floor scheduling using real-time information: A case study from the thermoplastic industry	✓	MILP formulation + CPLEX Solver + Simulated Annealing
Li et al., 2023	0	A Two-stage Stochastic Programming for AGV scheduling with random tasks and battery swapping in automated container terminals	✓	Two-stage Stochastic Programming + Sample Average Approximation + Ant Colony Optimization Algorithm
Duan e Wang, 2022	4	Robust scheduling for flexible machining job shop subject to machine breakdowns and new job arrivals considering system reusability and task recurrence		Dynamic Event Response Strategy (DERS) + Particle Swarm Arithmetic Optimization (PSAO)
Schlecht et al., 2022	0	Data-drive decision process for robust scheduling of remanufacturing systems	✓	Discreet Event Simulation + Robust Scheduling
Levorato et al., 2022	5	Exact solutions for the two-machine robust flow shop with budgeted uncertainty		MILP Problem + Column-and-Constraint Method
Novak et al., 2022	4	Scheduling jobs with normally distributed processing times on parallel machines		MINLP (Mixed Integer Non-linear Problem) + Branch and Prince Method
Sui e Wang, 2022	0	Parallel Identical Machines Scheduling to Minimize the Maximum Inter-completion Time with Uncertain Processing Time		Monte Carlo Simulation + K-Means Clustering + Genetic Algorithm
Panzer et al., 2022	0	Neural agent-based production planning and control: An architectural review		Neural Networks

Continua na página seguinte

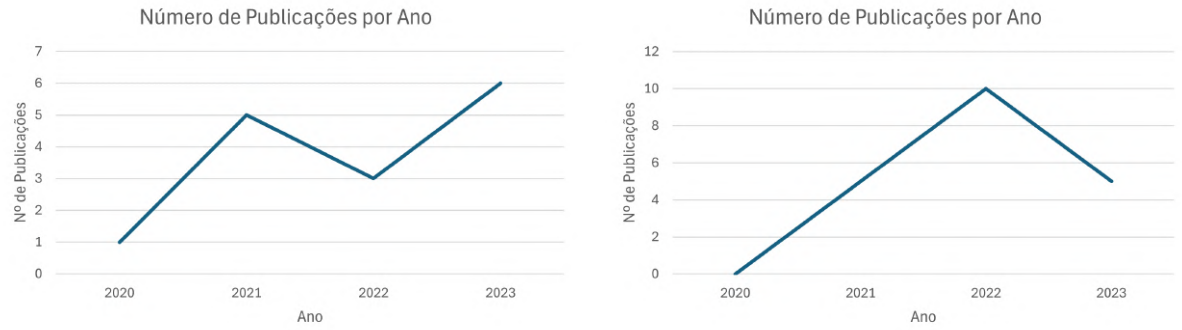
Referência	Cit	Título	EC	Métodos
Souza et al., 2022	16	Robust job-shop scheduling under deterministic and stochastic unavailability constraints due to preventive and corrective maintenance		MILP Problem + Genetic Algorithm
Hasan et al., 2022	0	A Rolling-Horizon Approach for a Surgery Case Scheduling Problem with Sterilizing Constraints	✓	MILP Problem + CPLEX Solver
Framinan et al., 2022	0	Assessing the potential of decentralised scheduling: An experimental study for the job shop case		MILP + Gurobi Solver
Manzini et al., 2022	2	A predictive–reactive approach for the sequencing of assembly operations in an automated assembly line	✓	MILP Problem + MATLAB Solver
Himmiche et al., 2021	2	A framework for robust scheduling under stochastic perturbations		Robustness Framework using PERT
Li et al., 2021	18	A green scheduling algorithm for the distributed flow-shop problem		INSGA II
Aramesh et al., 2021	14	A soft computing approach based on critical chain for project planning and control in real-world applications with interval data	✓	Critical Chain Method
Zimmermann et al., 2021	3	Multicriteria decision-making method for scheduling problem based on smart batches and their quality prediction capability	✓	AHP Method
Dehghan-Sanej et al., 2021	10	Solving a new robust reverse job shop scheduling problem by meta-heuristic algorithms		MIP + Simulated Annealing + Discreet Harmonic Search

2.1.2 Análise das publicações

É possível retirar conclusões sobre as tendências e padrões encontrados no conjunto de publicações analisadas, através da caracterização dos documentos pelo seu Ano de Publicação e relação entre *keywords*, através do *software VOSviewer*, na versão 1.6.19. Dado que existem dois conjuntos distintos de publicações, n_1 e $(n_2 + n_3)$, derivadas de temas e *strings* diferentes, estes serão analisados separadamente, com exceção da análise da relação entre *keywords*, para identificar potenciais áreas em comum entre ambos.

A caracterização dos documentos analisados pelo seu Ano de Publicação permite ter alguma noção sobre a dimensão da investigação a ser realizada nas áreas de interesse (Figura 2.2). Entre 2021 e 2022, a área de Veículos Autónomos viu um decréscimo do número de publicações, o que pode indicar a influência da pandemia COVID 19, por exemplo, no desenvolvimento de projetos de implementação deste tipo de tecnologias (Figura 2.2a). No entanto, a área da Calendarização Robusta viu um crescimento no número de publicações entre 2020 e 2022, o que indica um maior investimento na automatização de processos através da calendarização robusta de operações, demonstrando a intenção de académicos e do mercado em investir na melhoria

dos seus planeamentos, por influência da integração de tecnologias da Indústria 4.0 (Figura 2.2b). No ano 2023, as tendências inverteram-se, o que demonstra o retorno ao equilibro dos esforços entre a implementação de tecnologias de automação e o desenvolvimento de métodos de calendarização mais robustos, eficazes e eficientes.



(a) Veículos Autónomos (Conjunto n_1)

(b) Calendarização Robusta (Conjunto $n_2 + n_3$)

Figura 2.2: Número de publicações analisadas por ano de publicação.

O *software VOSviewer* é uma ferramenta que permite analisar dados bibliográficos através de mapas em rede e de densidade de um determinado tipo de dados, como por exemplo publicações, *journals*, autores e *keywords* (Van Eck & Waltman, 2010). A Figura 2.3 representa o mapa em rede das *keywords* que ocorrem, no mínimo, em quatro publicações. A distância entre duas *keywords* é indicativa do grau de afinidade entre duas *keywords*, dependendo do número de ocorrências nas publicações que partilham. O tamanho dos círculos está associado ao peso de cada *keywords* face ás restantes (van Eck & Waltman, 2022).

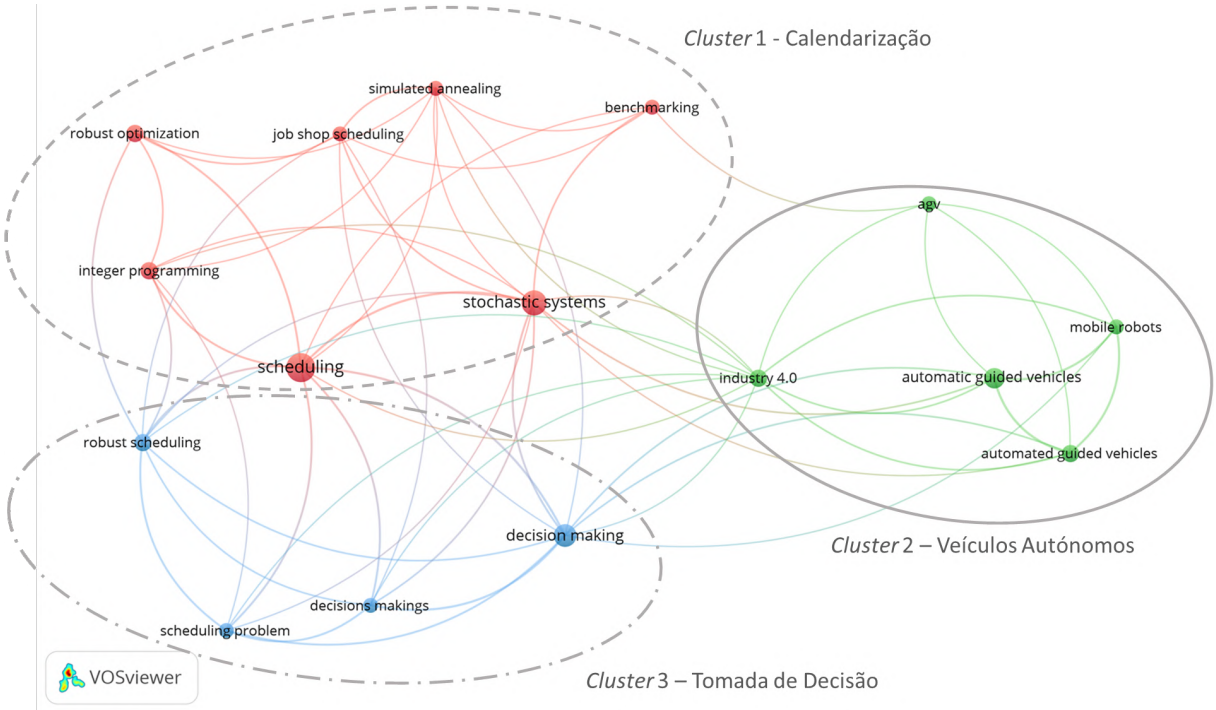
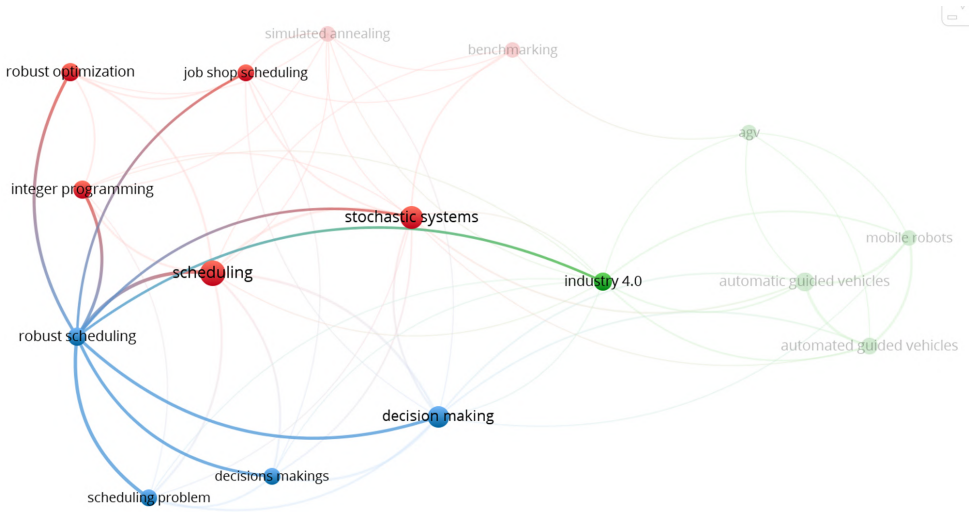


Figura 2.3: Mapa em rede das *keywords* de todas as publicações analisadas. Foram encontradas 23 *keywords* que ocorreram, no mínimo, em 4 publicações.

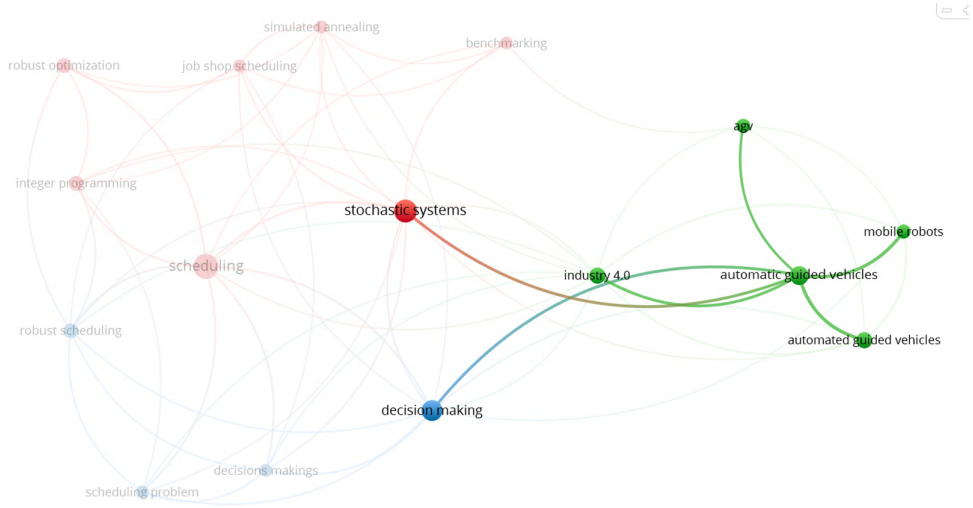
Identificam-se 3 *clusters*, associados aos temas de Calendarização, Veículos Autónomos e

Tomada de Decisão. Dada a quantidade de publicações relevantes na área da Calendarização Robusta, é natural que "*scheduling*" seja a *keyword* com maior peso. O *cluster* sobre Veículos Autónomos é o mais distante dos restantes, o que indica a menor afinidade deste tema com a Calendarização, principalmente na área da produção. No entanto, este *cluster* está ligado aos restantes através da *keyword* "*industry 4.0*", o que revela que tanto a temática de Veículos Autónomos como a Calendarização e Tomada de Decisão são focos de novas publicações e investimento, como indicado pela tendência crescente do número de publicações da Figura 2.2.

Em particular, na Figura 2.4, as *keywords* mais relevantes para o presente estudo são "robust scheduling" e "automatic guided vehicles" (escolhida pela dimensão do circulo face ás restantes).



(a) *Keyword "robust scheduling"*



(b) *Keyword "automatic guided vehicles"*

Figura 2.4: Mapa em rede das *keywords "robust scheduling"* e *"automatic guided vehicles"*.

Estas *keywords* têm afinidade com as *keywords "stochastic systems"*, *"industry 4.0"* e *"decision making"*, o que indica que a RS tem potencial para ser uma metodologia de sucesso no planeamento de tarefas de veículos autónomos, pelo enquadramento destes agentes em ambientes dinâmicos e com diversas fontes de incerteza. Verifica-se também o foco nas tecnologias da Indústria 4.0

para construir os modelos de calendarização e planeamento pretendidos, nomeadamente com possível recurso à Inteligência Artificial, *Machine Learning*, Digitalização de Processos e *Digital Twins* dos sistemas em que se integram.

2.2 Veículos Autónomos como Agentes Intralogísticos

Os AGVs e AMRs têm como principal objetivo o reabastecimento das unidades produtivas ou linhas de produção de uma fábrica, movimentando matérias-primas e componentes necessários sem necessidade de interação humana para realizar estas tarefas. O que diferencia estas dois tipos de veículos é a forma como navegam e transportam a carga entre dois pontos. Os AGVs, tal como o nome indica, são veículos guiados por um determinado mecanismo para seguir uma rota pré-definida. O primeiro AGV foi criado nos Estados Unidos, em 1953, pela empresa Barrett Electronics (Oyekanlu et al., 2020). Por outro lado, os AMR são veículos autónomos, capazes de determinar o caminho a percorrer de forma a realizar determinada tarefa, num local específico da instalação. A forma como o caminho é escolhido é um dos problemas a resolver e que é alvo de um estudo intensivo tanto pela empresa como em ambiente académico (Afonso et al., 2021; Gao et al., 2022; Mahmood et al., 2021; Reith et al., 2021; Stączek et al., 2021). Os AMR são mais utilizados em grandes armazéns ou fábricas, para transportar desde matérias primas a produto acabado (Stączek et al., 2021).

2.2.1 Principais desafios

Uma das características que mais diferencia os vários AGVs e AMR que existem e que deve ser escolhida conforme o ambiente em que se encontram é o tipo de navegação e sistema de controlo. Os veículos podem ser guiados através de lasers, sensores *wireless*, *barcodes*, RFID (*Radio Frequency Identification*), bandas magnéticas, sistemas de navegação ótico, ultrassónicos ou visuais e GPS (Fragapane et al., 2021; Oyekanlu et al., 2020). Outra característica a destacar são os sistemas de controlo de posição, que utilizam a localização e sensores do veículo para interagir com o ambiente que o rodeia e recolher dados para posterior simulação. Reis et al. (2023) compila vários sistemas de controlo, de entre os quais o controlo Clássico, Inteligente, Robusto, Preditivo, Ótimo, Moderno e Adaptativo.

A automação das operações de intralogística, através de AGVs/AMRs, incorre de vários desafios e problemas a resolver, de forma a aumentar a eficiência produtiva e reduzir custos operacionais:

1. A **calendarização dos veículos** para a satisfação das encomendas de material dentro da fábrica e a **definição de rotas** a percorrer dentro da instalação são dois grandes problemas e nos quais recaem grande parte dos esforços por parte da comunidade científica e dos peritos na área (Liu et al., 2022). Ambos têm grande impacto do aumento da eficiência das operações de logística interna e, por consequência, na produtividade.

   A calendarização ou sequência de tarefas a cumprir pelo veículo é afetada tanto pela variabilidade no tempo de carga e descarga como pela precisão da sequência criada através dos sistemas de planeamento (que utilizam dados em tempo real da linha ou unidade de produção) (Liu et al., 2022).

O planeamento de rotas dos veículos depende da forma como cada AGVs/AMRs está consciente do ambiente em que se encontra. Para além da clara necessidade de reduzir distâncias entre dois pontos da fábrica (Gao et al., 2022; Reith et al., 2021; Stączek et al., 2021), a gestão de conflitos entre veículos com rotas que se intercetam ou com obstáculos dinâmicos implica a definição a priori de como esses conflitos são detetados. Os AGVs são guiados por caminhos predefinidos, o que leva a rotas mais previsíveis e de trânsito significativamente controlado. Os AMR, por outro lado, são livres de determinar a sua própria rota e, por isso, são mais prováveis de encontrar outros veículos no seu caminho em *bottlenecks*, zonas centrais da fábrica ou *shortcuts* (Reith et al., 2021).

O teste de métodos e algoritmos de calendarização e roteamento de veículos autónomos pode ser feito a partir de dados recolhidos de um sistema real e no qual se pretende implementar o método desenvolvido. No entanto, nem sempre existe a possibilidade de implementar e testar estas ferramentas na realidade. Para colmatar esta falha, é possível construir *data sets* compreensivos e consistentes para testar novas abordagens de calendarização e roteamento de veículos. Sperling et al. (2023) compila a informação de 72 fluxos de material e *layouts* diferentes, juntamente com a respetivas matrizes de Transporte (cadência do abastecimento), matrizes de Distâncias, matrizes de movimentações em vazio, listas de postos e listas de tarefas. Para normalizar todos os fluxos e torna-los aptos à leitura pelos veículos autónomos, foi criada uma taxonomia para descrever os tipos de orientações de caminhos percorridos pelos veículos, tipos de *layout* e estrutura de tarefas a executar, bem como uma *framework* de recolha de dados para as matrizes de Transporte e de Distâncias. Até à data, é o único *data set* com informação consistente e normalizada para o teste de novos métodos e algoritmos, o que demonstra a necessidade de recolha e normalização de dados para o teste destas novas tecnologias (Sperling et al., 2023).

2. A **dependência energética** dos veículos é outro obstáculo à melhoria da eficiência das operações em causa. Geralmente, os veículos são alimentados através de baterias que precisam de ser carregadas a cada 8 a 10 horas. Portanto, a calendarização e definição de rotas ou mapas de caminhos possíveis deverá ter em conta não só a conceção de rotas energicamente mais eficientes, bem como a calendarização de tarefas que tenham em conta, por exemplo, o momento de carregamento dos veículos (Gao et al., 2022). É importante referir que Oyekanlu et al. (2020) compila diversos modelos e algoritmos utilizados para localização, calendarização, planeamento de rotas, comunicação, controlo e navegação de AGV/AMR nos mais variados contextos.

3. A **segurança** é outro dos desafios dos AGV e AMR. A velocidade com que se deslocam na instalação, a habilidade de se ajustarem ao contexto em que se encontram (perto de máquinas ou perto de operadores) e utilização de sinais visuais de aviso são algumas das estratégias a considerar na implementação destes veículos em contexto industrial. É necessário definir velocidades e distancias de travagem, bem como zonas exclusivas para AGVs/AMRs ou exclusivas para operadores, garantindo a segurança e bem estar dos seres humanos e equipamentos envolventes (Stączek et al., 2021). Chen et al. (2023) propõe um sistema de gestão de congestionamento de AMRs para agilizar a interação entre humanos e

agentes autónomos, que informa o utilizador sobre as rotas mais movimentadas e permite ao operador escolher a rota que menos interage com os robôs. Estes sistemas, no entanto, pressupõem que cada robô tenha o mesmo nível de conhecimento sobre o estado atual dos restantes agentes, mesmo que sejam de fornecedores diferentes, o que nem sempre é possível. Existe também a possibilidade de criar sinergias entre humanos e veículos autónomos, colocando os robôs como ferramenta do operador e não um agente totalmente independente. Sierra-García et al. (2023), por exemplo, desenvolveu um algoritmo de controlo de veículos autónomos que permite ao operador completar tarefas ergonomicamente complexas, como o manuseamento de objetos de diferentes tamanhos e pesos, ou a procura por objetos em *racks*. Qualquer AGV ou AMR que utilize esta tecnologia consegue reconhecer objetos espaçados de, no mínimo, 30 cm, o que resulta na correta deteção do operador.

4. A introdução de AGVs ou AMRs altera as rotinas de operadores e gestores, gerando **receios e resistências**, especialmente entre os operadores logísticos, que os veem como uma ameaça aos seus postos de trabalho. Nesse contexto, os gestores de projeto assumem um papel crucial na aceitação e satisfação dos operadores durante a implementação desta tecnologia. Através de comunicação clara e demonstração dos benefícios, os gestores de projeto podem ajudar a mitigar as preocupações dos operadores e facilitar a adaptação à sua nova realidade (Kopp et al., 2023). Inclusive, a identificação e mitigação de riscos durante a execução destes projetos contribuí para a aceitação e satisfação de operadores, gestores de projeto e gestores de topo, utilizando a FMEA (*Failure Modes and Effects Analysis* ou Análise de Modos de Falha e Efeitos), por exemplo, para identificar riscos internos e externos como fraca qualidade do produto final, possíveis paragens na produção e ainda colisões do veículos com operadores e outros equipamentos (Bekishev et al., 2023).

2.2.2 Indicadores de desempenho

Em sistemas já operacionais, em ambiente fabril, ou durante a simulação de linhas ou unidades de produção ainda por implementar, é necessário monitorizar e avaliar a performance da frota de veículos, tendo em conta, novamente, a melhoria da eficiência e da produtividade das operações logísticas. Para o efeito são utilizados *Key Performance Indicator* (KPI) que melhor caracterizam o quão eficiente é a carga, transporte e descarga dos materiais e componentes.

Os KPI "tempos de reposição" do material, "pontualidade do veículo" e "eficiência na descarga" estão relacionados com a operação de reabastecimento da linha ou unidade de produção. Dependem de avarias ou paragens de máquinas, encomendas de última hora ou alteração do tempo de entrega programado, chegada antecipada ao local de descarga ou reabastecimento, *buffers* de material cheios ou operadores indisponíveis para continuar o trabalho em espera (Liu et al., 2022).

A performance dos veículos também pode ser avaliada através do número de tarefas ou encomendas feitas por dia e do tempo médio de execução de uma encomenda. O número de incidentes (com outros veículos ou operadores) que o veículo incorre por dia afeta os KPI anteriores e reflete a necessidade de uma melhor adequação do meio envolvente ao trânsito destes veículos e/ou de um melhor planeamento de rotas (Mahmood et al., 2021; Stączek et al.,

2021). Pode também utilizar-se o custo por viagem ou transporte para garantir a sustentabilidade económica do sistema (Mahmood et al., 2021).

Um indicador que poderia ser incluindo, mas que não está presente em nenhuma das publicações analisadas, é o *Overall Equipment Effectiveness* (OEE). Este indicador resulta do produto dos indicadores "Qualidade", "Performance"e "Disponibilidade"e pode ser aplicado a apenas um elemento ou a um conjunto de elementos em estudo. Um OEE de 100% indica que determinado equipamento está sempre disponível, contribuí sempre com um *output* de qualidade e é eficiente e eficaz no desempenho da sua função (Pinto et al., 2019).

2.2.3 Simulação de Eventos Discretos

A implementação de frotas de veículos autónomos ou de outros sistemas mais ou menos complexos, em contexto real, necessita de uma representação digital do problema (*digital twin*) para testar os algoritmos e modelos propostos nas condições em que será implementado (Afonso et al., 2021; Mahmood et al., 2021; Stączek et al., 2021).

Segundo Law (2013), a *Discreet Event Simulation* ou Simulação de Eventos Discretos (DES) possibilita a representação da evolução de um sistema cujo estado varia em pontos discretos e distintos no tempo. Nem sempre é necessário construir um modelo de simulação, pois o sistema real pode já estar implementado. Caso o sistema real não exista, pode construir-se um modelo de simulação físico ou matemático e tão complexo quanto o decisor quiser, de acordo com a capacidade e recursos do momento. A DES insere-se no tipo de modelos estocásticos, dinâmicos e discretos, ou seja, modela sistemas onde os inputs incluem elementos estocásticos, a variável tempo é determinante para a obtenção de resultados e a evolução do sistema dá-se em pontos discretos no tempo. Para construir modelos de simulação válidos e estatisticamente relevantes, é necessário seguir um conjunto de etapas de simulação (Figura 2.5) e iterar qualquer fase que não tenha o resultado esperado.

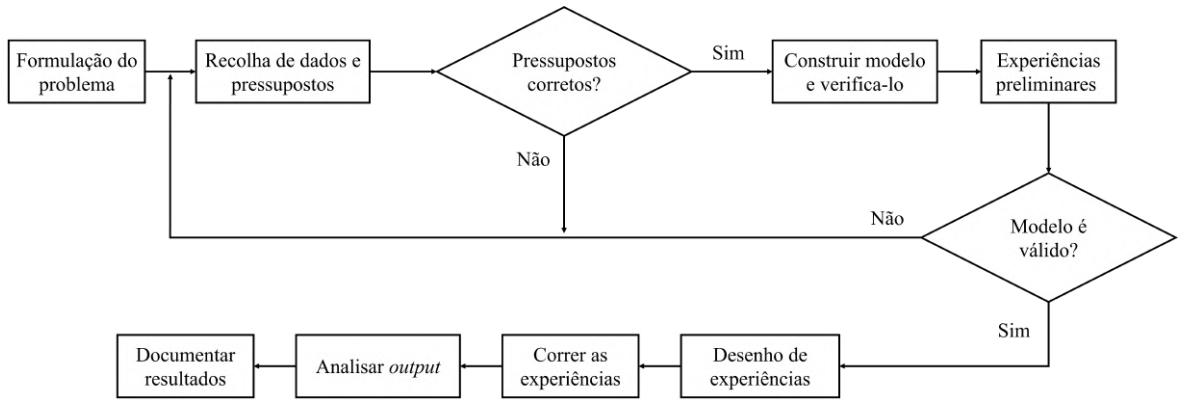


Figura 2.5: Etapas da DES. Adaptado de (Law, 2013, p. 67).

É comum utilizarem-se softwares como o FlexSim, Simio (Afonso et al., 2021) ou Gazebo (Stączek et al., 2021), para verificar e validar os modelos propostos, assim como garantir que os KPI desejados são atingidos.

No que toca ao uso da DES na implementação de frotas de AGV/AMR, esta pode e deve ser utilizada no teste de modelos e algoritmos de calendarização e roteamento, entre outros. Uma limitação ds DES é a necessidade de modelos validos do sistema onde se pretende testar tais

abordagens. Na presença de modelos já validados, a DES é uma mais valia na construção de modelos robustos, capazes de superar diversos cenários com vários graus de variabilidade e incerteza (Schlecht et al., 2022).

Esta ferramenta permite também o trabalho em equipa num mesmo ambiente e impede o investimento inicial desnecessário, caso o sistema não seja adequado à situação em estudo (maior flexibilidade antes da implementação) (Stączek et al., 2021).

2.3 Calendarização Robusta (RS): Uma Reflexão

Como mencionado anteriormente, um dos grandes desafios na implementação de frotas de AGV e AMR em chão de fábrica é a calendarização das tarefas a desempenhar pelos veículos. Focando a pesquisa na RS, esta é caracterizada pela sua adaptabilidade a ambientes voláteis e sujeitos a eventos imprevisíveis, tais como avarias de máquinas ou veículos, tempos de ciclo bastante variáveis e encomendas de emergência (Duan & Wang, 2022).

Em ambientes de produção tão dinâmicos como, por exemplo, a indústria automóvel, a constante renovação da tecnologia produzida impõem a necessidade de rápida adaptação e flexibilidade dos recursos disponíveis ás alterações da linha de produção e montagem. Por outro lado, a Indústria 4.0 veio revolucionar a forma como interagimos com o ambiente fabril, a forma como comunicamos com a linha e, principalmente, o tipo e quantidade de informação que conseguimos obter em cada instante. Assim, é necessário convergir para Sistemas Ciberfísico, flexíveis, robustos a imprevistos, rapidamente adaptáveis a novos processos e com a capacidade de recolha de informação sobre o estado atual do sistema e as suas necessidades a nível operacional (Framinan et al., 2022).

Esta troca de informação entre os vários agentes reflete-se num novo problema de calendarização de operações, alvo de grandes esforços de investigação na área da Investigação Operacional. Este pretende incorporar as perturbações que um sistema está destinado a encontrar, tornando-o mais eficiente e apto a superar possíveis desvios em relação ao planeado. Por isso, este tipo de problemas estão normalmente divididos em duas questões (Himmiche et al., 2021):

1. Encontrar uma solução ótima de calendarização;
2. Avaliar a sua *performance* na presença de perturbações.

Nos últimos anos, uma nova vertente destes problemas surge com a tentativa de incorporar a sustentabilidade ambiental e determinar uma solução de calendarização que pretenda não só reduzir custos operacionais e utilização de recursos escassos, mas também que seja igualmente ecológica (Li et al., 2021).

Para determinar a melhor solução, ou seja, o melhor calendário, são usadas várias formulações para este tipo de problemas, bem como vários algoritmos e *solvers* que selecionam a melhor solução de entre um conjunto de soluções possíveis. A Programação Linear Inteira Mista é uma das formulações mais utilizadas entre os peritos, pela sua facilidade e relativa simplicidade face a outros métodos, tais como:

- Método do Caminho Crítico (Aramesh et al., 2021; Peng et al., 2023)

- Algoritmos Genéticos (Souza et al., 2022; Sui & Wang, 2022);
- Redes Neuronais (Panzer et al., 2022);
- Programação Inteira Mista Não Linear (Novak et al., 2022);
- Estratégia de Resposta a Eventos Dinâmicos (DERS) (Duan & Wang, 2022);
- Processo de hierarquia analítica (AHP), um método de decisão multi-critério (Zimmermann et al., 2021);
- INSGA II, um algoritmo de otimização multi-objetivo (Li et al., 2021);

Note-se que todos estes métodos foram identificados nas publicações indicadas na Tabela 2.1, que agrega as publicações mencionadas no presente estado da arte.

Nesta secção são sintetizados vários tipos de problemas mais frequentes, objetivos, medidas de desempenho e algoritmos que permitam obter uma solução de calendarização ótima ou sub-ótima.

2.3.1 Problemas mais comuns

A RS pretende construir um planeamento das operações a desempenhar por um conjunto de entidades e que tenha em conta eventos dinâmicos e imprevisíveis que interferem com a sua execução. Os problemas de calendarização em publicações mais recentes focam-se na área do planeamento de produção e manutenção, não havendo estudos direcionados à calendarização robusta de veículos autónomos, em ambiente industrial. Os seguintes problemas encontram-se totalmente descritos, por exemplo, na obra *"Scheduling: Theory, algorithms, and systems"*, de Pinedo (2016), onde as abordagens determinística e estocástica são estudadas pelo autor.

2.3.1.1 Calendarização em *Flow Shop* Permutacional

O *Permutation Flow Shop Problem* ou Problema de Permutação de *Flow Shop* (PFSP) é definido como um conjunto de n encomendas a processar por um conjunto de m máquinas, numa ordem pré-determinada. Cada encomenda é processada por cada máquina e colocada numa fila de espera (normalmente utilizando o método *First In First Out* (FIFO)) para aguardar o início do processo de maquinação seguinte. Isto significa que cada encomenda deverá ser programada em cada máquina para completar o processo de fabrico. Como tal, o problema pode ser extrapolado para qualquer conjunto de tarefas que tem de ser executado por cada membro de outro conjunto (Pinedo, 2016).

É uma configuração bastante usada em linhas de produção/montagem nas mais variadas indústrias: química, petroquímica, automóvel, metalúrgica e alimentar. Dado que se tratam de fluxos contínuos, é imperativo manter o fluxo de *Work in Progress* (WIP) em movimento, pelo que é comum minimizar-se os tempos de espera entre processos, de forma a aumentar a cadência de produção e lucros (Levorato et al., 2022; Manzini et al., 2022). Um exemplo deste tipo de problema reside no planeamento das operações de manutenção em poços de petróleo e gás, onde as tarefas têm uma sequência pré-definida e devem ser realizadas em todos os poços de cada plataforma (Levorato et al., 2023).

A generalização do problema PFSP é denominado problema de *Flow Shop* Permutacional Distribuído, onde é mantida a configuração de *flow shop*. No entanto, além de ser considerada a sequência de tarefas a realizar, é necessário alocar tarefas a um conjunto de fábricas. É comum utilizar-se a minimização do *makespan*, embora várias publicações se foquem na otimização multi-objetivo, ou seja, num conjunto de funções objetivo que se pretende otimizar (Li et al., 2021).

2.3.1.2 Calendarização de Máquinas Paralelas

Um *Parallel Machine Problem* ou Problema de Máquinas Paralelas (PMP) pode ser caracterizado como um conjunto de m máquinas idênticas para as quais é necessário alocar um conjunto de n trabalhos. A conclusão destas tarefas não tem restrições específicas de ordem ou precedência e podem ser calendarizadas com ou sem interrupção das mesmas (Pinedo, 2016). Da mesma forma, um conjunto de n tarefas pode ser alocado aos elementos de um conjunto de m entidades, sem restrições de precedência ou sequência de processamento definida, desde que cada tarefa seja executada por apenas uma entidade. As aplicações deste problema variam desde o ambiente de fabrico até à computação na nuvem (*Cloud Computing*), gestão de projetos, operações em estaleiros navais, portos e consultas de doentes em hospitais e clínicas (Cohen et al., 2023).

Este tipo de problema é muito utilizado para a alocação de tarefas que requerem mão-de-obra humana, cujo tempo de processamento pode ser definido por uma distribuição normal, ou seja, a duração das tarefas são estocásticas. Estão geralmente sujeitos a metas *end-of-the-day*, para atingir cotas de produção. Portanto, uma solução ótima é um calendário que maximiza a probabilidade de todas as tarefas serem cumpridas até do prazo pré-estabelecido (Novak et al., 2022). É interessante a formulação destes problemas com a introdução de tarefas urgentes pois, juntamente com os tempos de processamento estocásticos, é uma representação mais realista do que realmente ocorre no mundo real. Nestes casos, é minimizado o máximo tempo de espera de uma tarefa urgente, recorrendo a um conjunto de planeamentos e a algoritmos que selecionam a solução ótima (Sui & Wang, 2022). O planeamento de cirurgias, por exemplo, pode ser considerado um PMP, pois os blocos operatórios são considerados recursos idênticos, aos quais deverão ser alocadas cirurgias de duração probabilística. É comum existirem tarefas não urgentes e urgentes, que correspondem a cirurgias eletivas e não eletivas, respetivamente (Hasan et al., 2022).

2.3.1.3 Calendarização de *Job Shop* Flexível

Um *Flexible Job Shop Problem* ou Problema de *Job Shop* Flexível (FJSP) é um problema que pode ser descrito como sendo semelhante ao PMP, mas, ao invés de ter m máquinas idênticas em série, estas estão divididas em centros de trabalho diferentes, com máquinas idênticas. Cada encomenda passa por cada centro de trabalho e só pode ser programada para uma máquina em cada um (Pinedo, 2016). Este é um cenário muito mais realista, uma vez que, em ambiente de produção, existem várias máquinas com operações específicas e semelhantes, formando grupos de máquinas para cada operação de maquinação (Duan & Wang, 2022; Souza et al., 2022).

O FJSP inverso é uma generalização do problema de *Job Shop*, onde são considerados fluxos inversos (de reprocessamento, por exemplo) e cujas tarefas devem ser incluídas no

planeamento para otimizar a medida de desempenho escolhida (geralmente o *makespan*). Este tipo de problemas pode ser aplicado nas indústrias automóvel, de eletrodoméstico e eletrónica, onde existem linhas de montagem/desmontagem para diversos produtos (Dehghan-Sanej et al., 2021).

2.3.2 Classificação e Notação

Os problemas de calendarização anteriores, bastante usados em ambientes de produção, são descritos usando várias formulações matemáticas que tentam encapsular as principais características do problema. Em particular, os problemas MILP têm uma estrutura comum que prevalece em todas estas tipologias de problemas e nos quais a função objetivo é uma das componentes mais importantes. Simultaneamente, as características do ambiente de produção têm um papel importante na escolha da formulação correta do problema, pelo que a sua representação correta é obrigatória. O tipo de planeamento construido, a forma como é implementado e se pode ou ser ser atualizado enquanto o sistema está em funcionamento são também informações importantes para melhor compreender o problema MILP a resolver.

Duas abordagens frequentemente utilizadas para alocar tarefas a recursos são a Calendarização de Alocação Estática e Calendarização de Lista Estática. A Calendarização de Alocação Estática atribui cada tarefa a um recurso específico. A Calendarização de Lista Estáticas cria uma sequência de tarefas e cada tarefa é atribuída assim que um recurso está disponível (tendo em conta que, à partida, os recursos são considerados independentes e idênticos e/ou que o recurso é escolhido segundo uma heurística pré-determinada). Ambos beneficiam da minimização/maximização da função objetivo, mas são executados de forma diferente (Cohen et al., 2023).

Existem também três abordagens para a criação ou atualização de um planeamento enquanto o sistema está a funcionar:

- Reativa ou Adaptativa - Gera um novo planeamento à medida que nova informação sobre o sistema é disponibilizada;
- Proativa ou Preditiva - Cria um novo planeamento com dados históricos sobre o sistema e não se adapta à medida que nova informação é disponibilizada;
- Proativa-Reativa ou Preditiva-Adaptativa - Um primeiro planeamento é gerado através de dados históricos e com a informação disponível no momento. Novos planeamentos são depois geradas/alteradas à medida que é conhecida mais informação sobre o sistema. Esta é a prática mais comum em problemas de calendarização, dada a sua flexibilidade e resiliência a alterações do ambiente envolvente (Cohen et al., 2023; Ghaleb & Taghipour, 2023; Himmiche et al., 2021; Manzini et al., 2022).

Quando combinados modelos de Otimização Robusta com a calendarização adaptativa, que integram decisões futuras ótimas na fase de planeamento, superam os modelos de Otimização Robusta não adaptativos, uma vez que minimizam o tempo de execução máximo possível do programa. Os modelos adaptativos não só fornecem melhores previsões de *makespan*, como

também resultam numa melhor duração real das tarefas quando o c é cumprido. Quando comparados com modelos totalmente reativos, os modelos proativos-reativos têm um desempenho tanto pior quanto mais decisões proativas tomarem no primeiro planeamento gerado (Cohen et al., 2023).

Problemas do tipo PMP beneficiam de modelos adaptativos de Otimização Robusta quando consideram incerteza moderada, em vez de cenários de pequena ou elevada incerteza. Quando existe uma fase proativa, o desempenho do algoritmo piora quanto mais decisões de planeamento forem tomadas na fase proativa, o que confirma o desempenho superior dos modelos adaptativos puros, que têm em conta a informação à medida que o calendário é cumprido e mais calendários são criados (numa abordagem *Rolling Horizon*) (Cohen et al., 2023; Hasan et al., 2022). A título de exemplo, no caso da indústria de termoplásticos, a utilização de algoritmos como *Simulated Annealing* para avaliar um conjunto de planeamentos gerados a partir da formulação MILP do problema, à medida que o sistema evolui, reduz o custo total ponderado em 6,99% e, como consequência, leva à redução do custo por atrasos na produção em 26,1%. Isto pode ser traduzido em ganhos económicos, bem como na manutenção de um bom relacionamento com o cliente, que se traduz em lealdade ao atual fabricante (Cohen et al., 2023). A escolha da solução ótima num conjunto de planeamentos geradas, em alguns casos, reduz igualmente a necessidade de horas extraordinárias, ocupação dos recursos utilizados e contribui para menores níveis de stress por parte dos recursos humanos utilizados (Hasan et al., 2022). A adaptação do processo de planeamento ao ambiente volátil que o rodeia cria um processo de tomada de decisão autónomo, que se adapta a perturbações e que inclui não só um modelo de calendarização robusto, como uma rede de entidades que permitem a recolha de informação em tempo real (Himmiche et al., 2021).

Todos os métodos e abordagens de sequenciação e calendarização de operações necessitam de uma notação adequada, que seja facilmente reconhecível e que resuma as características determinantes do problema em questão (Ghaleb & Taghipour, 2023; Levorato et al., 2022, 2023). A notação $\alpha|\beta|\gamma$ foi introduzida por Graham et al. (1979) com o intuito de classificar problemas de calendarização determinística através dos seguintes campos:

- α - Características das máquinas, da produção ou recursos no geral, como os tipos de problema (PMP ou FJSP, por exemplo);
- β - Características das tarefas, como relações de precedência e existência ou não de *preemptive tasks*;
- γ - Função objetivo para minimizar/maximizar.

Este tipo de notação é clarificada por Graham et al. (1979), onde todas as opções para os campos α, β e γ são explicadas ao detalhe, para que os leitores posso escolher quais as características que melhor se adequam ao seu problema.

No âmbito da calendarização robusta, onde existem fontes de incerteza que pretendem simular os eventos aleatórios do mundo real, a calendarização determinística dá lugar à calendarização estocástica, que inclui esses mesmos fatores (Pinedo, 2016).

2.3.3 Funções Objetivo

A Programação Linear é o ramo da Investigação Operacional que se foca na alocação de recursos limitados a atividades, de forma a otimizar determinado objetivo através do planeamento ótimo proposto, admitindo a utilização de **funções lineares**, ou seja, somatório de termos compostos por **variáveis de decisão** e **parâmetros**, cujo número e natureza dependem do problema em questão. Existirão tantas variáveis de decisão quanto o número de decisões quantificáveis a tomar. A medida de desempenho que se pretende otimizar, isto é, maximizar, minimizar ou igualar é descrita na **função objetivo**, que inclui todas as variáveis de decisão. Dado que o problema pressupõe recursos limitados, as variáveis de decisão são limitadas por um conjunto de **restrições** na forma de inequações ou equações lineares (Hillier & Lieberman, 2015).

A escolha da melhor medida de desempenho do planeamento, ou seja, da melhor função objetivo, é determinante para a obtenção da solução que mais beneficia o decisor. Embora as restrições relatem a realidade dos recursos disponíveis no momento, é a função objetivo que dita quais as decisões a tomar para os utilizar da forma mais eficaz e eficiente possível.

A função escolhida irá depender do tipo de problema em questão e da área a que se aplica. Na calendarização de manutenção preventiva em poços de petróleo, por exemplo, onde o planeamento se enquadra nos problemas do tipo PFSP, a necessidade de reduzir o tempo que determinada operação está parada para manutenção leva a adoção da função objetivo Tempo Total Ponderado de Conclusão (Levorato et al., 2023). Desta forma, a minimização do tempo de conclusão de todas as operações de manutenção leva à satisfação do nível de serviço pretendido, à redução dos custos de produção e à manutenção da qualidade, com um produto final mais uniforme (Manzini et al., 2022). Problemas do tipo PFSP podem incluir também funções objetivo que permitam maior eficiência energética ou que minimizem, por exemplo, o tempo total em fluxo (que contempla tanto tempo de processamento, como tempos de espera e de transporte do lote em produção), o consumo total de energia (Li et al., 2021), o *delay* máximo ou o WIP no sistema (Himmiche et al., 2021).

Em problemas do tipo PMP, onde várias operações têm que ser executadas em máquinas semelhantes, pretende-se diminuir o tempo de produção da sequência tarefas ou *makespan*, criando desta forma uma calendarização com uma distribuição de trabalho equilibrada entre máquinas, mesmo existindo a possibilidade de interromper tarefas durante a sua execução (denominadas *preemptive tasks*) (Cohen et al., 2023; Ghaleb & Taghipour, 2023). A minimização do makespan é um problema *NP-hard* para 3 ou mais máquinas. Na versão do problema com duas máquinas, o *makespan* ótimo é o mesmo, com ou sem a restrição de permutação entre máquinas (Levorato et al., 2022). Outros autores sugerem incorporar direta ou indiretamente os custos de produção na função objetivo, de forma a minimizar cada um dos custos, incluindo energéticos (Duan & Wang, 2022). É também prática comum utilizar várias funções objetivo para a minimização do *makespan* e que contém diferentes parâmetros para o mesmo problema. Por exemplo, o impacto da calendarização da manutenção poderá ser estudado utilizando 3 funções objetivo para a minimização do *makespan*: *makespan* sem operações de manutenção, apenas com manutenção preventiva e ainda com manutenção preventiva e corretiva. Este último já incorpora falhas como eventos imprevisíveis, cuja duração e instante de início podem ser determinísticos ou estocásticos (Souza et al., 2022). Em problemas relacionados com a calendarização de AGVs

em particular, menos comuns na literatura, podem encontrar-se funções objetivo que pretendem minimizar o custo de falha na entrega e atrasos e custo do tempo de espera do veículo e da tarefa a desempenhar (Li et al., 2023).

2.3.4 Fontes de Incerteza

A RS tem o objetivo de absorver e acomodar fontes de variabilidade e incerteza que influenciam o funcionamento expectável de uma linha de produção ou de processos intralogísticos e criar um planeamento que reduz, por exemplo, os custos de produção, o tempo de conclusão ou o *makespan*. Eventos imprevisíveis, como durações de tarefas estocásticas, falhas de máquinas ou outras entidades, chegada de novas encomendas ou trabalhos, entrada de encomendas urgentes, cancelamento de encomendas e falta de matéria prima, afetam a estabilidade do sistema e do plano inicial (Levorato et al., 2022, 2023). Ações de manutenção preventiva e corretiva, em particular, têm grande impacto no output de um sistema, pois, em ambos os casos, as entidades intervencionadas ficam indisponíveis para desempenhar a sua função. Neste caso, é especialmente importante que a calendarização seja o mais robusta possível a este tipo de eventos (Manzini et al., 2022; Souza et al., 2022). Todas estas fontes de incerteza podem ser resumidas a 3 categorias de incerteza (Li et al., 2023):

- Fontes de incerteza externas, como o momento de chegada de uma encomenda ou a chegada de uma encomenda urgente;
- Fontes de incerteza internas, inerentes a operações, como a duração das tarefas a desempenhar;
- Anormalidades pontuais, como a necessidade de operações de manutenção corretiva;

Um dos métodos para absorver a variação inerente de um sistema real é a utilização da chamada *Budgeted Uncertainty*. Esta impede que se seja demasiado conservador no planeamento e fornece uma gama de possíveis calendarizações, que variam consoante a aversão ao risco do decisor e a tolerância do sistema a diferentes níveis de incerteza (Levorato et al., 2022). O decisor deverá escolher até que ponto o calendário deve ser conservador em relação à incerteza, ponderando, por exemplo, o desvio-padrão do tempo de processamento e calculando o desvio máximo do tempo de processamento permitido (Levorato et al., 2023).

A abordagem de Otimização Robusta, onde vários planeamentos são gerados e é escolhido aquele com a melhor valor da função objetivo, tem como fontes de incertezas mais comuns a duração das tarefas e a dependência entre a duração das mesmas. Outros problemas semelhantes incluem também a incerteza nos tempos de preparação, nas avarias de máquinas e na duração de determinados processos em cada máquina (preparação de moldes de injeção na indústria de termoplásticos, por exemplo). A manutenção corretiva é, novamente, outra fonte de incerteza na sua duração e instante de ocorrência (Ghaleb & Taghipour, 2023). Este tipo de problema utiliza uma função objetivo diferente, onde é minimizado o máximo *makespan* de cada planeamento (método *Minmax*) (Cohen et al., 2023; Levorato et al., 2022). Entre planeamentos, os parâmetros do problema são diferentes e, com a minimização do *makespan* máximo, garante-se que os melhores parâmetros também são escolhidos. No final, a solução são planeamentos robustas, com os

melhores parâmetros possíveis, garantindo planeamentos resilientes a eventos imprevistos, no pior cenário gerado. Esta combinação dos problemas MILP e Otimização Robusta é também designada por Otimização Linear Inteira Mista ou MILO (Cohen et al., 2023).

A RS não é apenas utilizada no setor da transformação de matérias primas em produtos de valor acrescentado. No setor terciário, os serviços de saúde beneficiam das vantagens que este tipo de planeamento de operações pode oferecer. A calendarização de cirurgias em ambiente hospitalar é outra área onde a incerteza tem grande influência na *performance* do sistema e, neste caso, na saúde dos utentes. Um dos grandes problemas é a estimativa da duração das cirurgias pois, embora existam dados históricos que permitem fazer uma estimativa, existe ainda uma grande diferença entre os valores previstos e as durações que realmente ocorrem, afetando a *performance* do planeamento inicial. Utilizando novamente o desvio-padrão para calcular a folga a adicionar no fim de cada cirurgia, é possível colmatar algumas destas falhas. Dado que esta calendarização é feita com semanas de antecedência e utilizando o método *Rolling Horizon* ou *Rolling Window*, os calendários são ajustados a cada semana para garantir que as cirurgias seguintes não são afetadas por atrasos no presente (Hasan et al., 2022).

Existem alguns fatores que dificultam a inclusão de fontes de incerteza na obtenção de planeamentos robustos. Numa linha de produção, a manufatura de pequenas séries de produtos, rápidas mudanças na sequências de processos ou sucessivos ajustes em máquinas durante a produção dificultam a criação de novos calendários ótimos. Existem também fontes de variabilidade nos processos que estão associadas a outras causas que não a variabilidade inerente do processo de transformação, como é o caso da necessidade de reprocessamento de unidades para fins de qualidade, cujas durações podem não estar assinaladas aquando da recolha de dados sobre o processo (Framinan et al., 2022). Esta falha na integridade e recolha de dados para a estimativa de tempos necessários à RS deriva do desafio na implementação destes Sistemas Ciberfísicos capazes de recolher dados em tempo real e de gerar dados suficientes e relevantes ao problema. As tecnologias da Indústria 4.0, embora em ascensão, não são ainda prática comum, mas que são necessárias à implementação deste tipo de planeamento (Himmiche et al., 2021).

2.3.5 Medidas de Desempenho da Robustez

Após a escolha da função objetivo a usar e da natureza dos dados de *input* no modelo, os planeamentos gerados por qualquer métodos de calendarização devem ser avaliadas quanto à sua habilidade para cumprir prazos de conclusão das tarefas e de ajustar as tarefas seguintes a quaisquer atrasos ou imprevistos que posso ocorrer.

A flexibilidade e adaptabilidade de um planeamento gerado e aceite como solução do problema tem duas componentes que devem ser avaliadas (Cohen et al., 2023):

1. A habilidade de calendarizar tarefas tendo em conta possíveis eventos futuros;

2. A habilidade para alterar a alocação de tarefas à medida que o sistema evolve.

Assim, podem identificar-se dois aspetos em que um calendário ou método de calendarização pode ser avaliado (Duan & Wang, 2022; Souza et al., 2022):

- Robustez de Qualidade ou Sensibilidade, que avalia a insensibilidade do calendário quando sujeito a incerteza (inclui indicadores com *makespan* ou atraso);
- Robustez de Solução ou Estabilidade, que avalia a habilidade do planeamento manter uma certo nível de qualidade na presença de incerteza (relacionado com desvios entre valor atual de um KPI e limite imposto).

A Robustez de Qualidade ou Sensibilidade pretende avaliar a sensibilidade do planeamento a perturbações no sistemas. Planeamentos mais sensíveis veem a sua qualidade diminuir ao longo do tempo, à medida que imprevistos ocorrem, pois desvios em relação ao planeado implicam atrasos significativos nas tarefas seguintes. São necessários vários ajustes do planeamento ao longo do tempo para manter os KPI nos níveis pretendidos. A Robustez de Solução ou Estabilidade refere-se à capacidade que o calendário tem de manter a qualidade do planeamento, ou seja, de se manter inalterado durante mais tempo e manter a sua performance ao longo do tempo (Himmiche et al., 2021).

Para avaliar a robustez de um calendário, é necessário perceber se e como este se altera ao longo do tempo. A forma como um planeamento é alterado depende da estratégia escolhida, ou seja, se os calendários são gerados de forma Preditiva, Adaptativa ou Preditiva-Adaptativa. Nestas ultimas duas abordagens, sempre que é gerado um primeiro calendário, este está sujeito a alterações com a chegada de nova informação sobre o sistema. Estas alterações têm como pressuposto a possibilidade de identificar atrasos no planeamento em curso assim que é detetada a conclusão de uma tarefa após o instante de termino planeado (Manzini et al., 2022).

Existe várias estratégias para alterar os calendários subsequentes. Quando ocorrem avarias, por exemplo, apenas as tarefas que ainda não foram iniciadas são alteradas e é gerado um novo calendário. Com a chegada de novas tarefas, as tarefas já calendarizadas não são alteradas. Desta forma garante-se que não existem muitas alterações no planeamento, principalmente quando este já está em execução (Duan & Wang, 2022). As alterações do planeamento também podem ser geridas por atrasos em relação ao planeamento inicial. Caso a ocorrência de um evento imprevisto não cause atrasos na calendarização, a alocação de tarefas a recursos não é alterada. No caso da ocorrência de atrasos, é despoletada uma reação para avaliar o calendário atual e perceber se é necessária nova otimização (Manzini et al., 2022).

Com estes mecanismos de alteração do planeamento implementados, é possível avaliar um método de calendarização ou um calendário apenas através de medidas de desempenho da robustez. A definição está normalmente associada à probabilidade de desvio de determinada medida face a um valor referência, avaliando o risco de perda de qualidade do planeamento ao longo do tempo. Por exemplo, a robustez pode ser definida como a probabilidade de determinado KPI ser menor ou igual ao limite definido pelo decisor (Himmiche et al., 2021). Se porventura existirem vários KPI, são analisados os níveis de robustez de todos as medidas e é escolhido o calendário ou método de calendarização cujo níveis de robustez são mais elevados (Schlecht et al., 2022).

Dada a variedade de definições de robustez que podem haver, principalmente por estarem dependentes dos KPI utilizados, é possível identificar alguns exemplos concretos de como esta é medida. Utilizando os desvios ocorridos na medida de desempenho utilizada como função objetivo é possível medir a robustez através do Desvio Relativo Percentual (RPD) (equação 2.1):

$$\mathrm{RPD}_{im} = \frac{\mathrm{C}^{im}_{max} + \mathrm{C}^{ir}_{max}}{\mathrm{C}^{ir}_{max}} \times 100 \tag{2.1}$$

Neste exemplo é utilizado o *makespan* C e onde se pretende avaliar o desvio relativo do *makespan* do planeamento i quando gerada pelo método m. Este é comparado um valor de referência C^{ir}_{max} para o planeamento i. Podem depois agregar-se todos os planeamentos gerados utilizando o desvio relativo do *makespan* médio e comparar esta medida entre métodos de calendarização, de forma a escolher aquele cujo desvio é, em média, mais próximo dos valores de referência. Os valores de referência podem ser iguais, por exemplo, ao valor objetivo obtido através abordagem determinística do problema, resolvido utilizando MILP. Dado que, geralmente, se pretende minimizar o *makespan*, pode ser utilizado também o valor mínimo de *makespan* obtido no planeamento i quando aplicado o método m (Framinan et al., 2022).

O nível de serviço (*Robustness Level* ou RL) do planeamento pode ser outra opção para avaliar a robustez. Utilizando uma perturbação do planeamento (*pert*), de natureza probabilística, o nível de serviço RL é definido por Himmiche et al. (2021) como a probabilidade de que o *makespan* C_{max} do planeamento s seja inferior ou igual ao prazo $\widetilde{d}$, considerando a perturbação *pert* com determinado nível de confiança (equação 2.2). Este nível de serviço deverá ser inferior ao limite estabelecido pelo decisor para concluir que o método gera calendários robustos.

$$RL(s, pert, \widetilde{d}) = Pr(\mathrm{C}_{max}(s, pert) \leqslant \widetilde{d}) \tag{2.2}$$

Utilizando a equação 2.2, o autor constrói 8 problemas de robustez que pretendem responder a diversas questões, nomeadamente qual o prazo $\widetilde{d}$ mínimo que garante o nível de serviço pretendido (Himmiche et al., 2021, p. 3).

2.3.6 Procurar a melhor solução: Algoritmos e *Solvers*

Quando formulamos um problema de calendarização usando MILP, são necessários métodos, algoritmos e *solvers* que possam fornecer uma solução ótima ou sub-ótima. Um *solver* é uma ferramenta onde é formulado o problema, utilizando uma determinada linguagem de programação. O objetivo é resolver o problema em questão com recurso a métodos ou algoritmos.

Os *solvers* mais usados entre académicos e organizações são o Gurobi e o IBM ILOG CPLEX Optimizer, dada a facilidade de obtenção de licenças para académicos e estudantes, bem como a recomendação para uso em softwares de simulação como o Flexsim (Software, 2023). Existem ainda outros *solvers* mencionados na literatura, como o BONMIN (Souza et al., 2022), aplicado através da plataforma GAMS.

Através destes *solvers* e usando vários métodos, é possível resolver diversos problemas formulados usando Programação Linear, em especial problemas MILP. O Método de Geração de Colunas e Restrições (formulado com os *solvers* CPLEX e Gurobi), por exemplo, consegue resolver problemas com até 15 tarefas a programar, embora sacrifique o tempo de processamento. Nestes casos, é prática comum utilizar um *cutoff time* para limitar o tempo gasto em cada planeamento testado, alcançando uma solução ótima em 48% das planeamentos testados, com as restantes soluções a apresentarem uma diferença de 5% em relação ao valor objetivo ótimo (Levorato et al., 2023).

Os problemas PMP, utilizando uma formulação MILP, usam o *solver* CPLEX (programado em Python ou MATLAB) para gerar planeamentos (Dehghan-Sanej et al., 2021; Ghaleb & Taghipour, 2023; Hasan et al., 2022; Manzini et al., 2022). Quando são geradas vários calendários, são aplicados outros algoritmos para escolher a melhor solução, ótima ou sub-ótima, como o *Simulated Annealing*. Neste caso, é necessário afinar os parâmetros do algoritmo, utilizando uma amostra de planeamentos diferentes, de diferentes instâncias. utilizadas para avaliar o modelo (Ghaleb & Taghipour, 2023). Neste contexto, instância é uma versão do problema original com diferentes parâmetros. *Design of Experiments* ou Desenho de Experiências pode ser utilizado neste caso para analisar as combinações de parâmetros que mais beneficiam a função objetivo. A gama de valores a estudar ultrapassa os 2 níveis por fator ou parâmetro, pelo que são necessárias metodologias de Desenho de Experiências mais complexas (Ghaleb & Taghipour, 2023).

Um método muito utilizado é o método de *Branch-and-Bound*(Levorato et al., 2023), cuja performance é ultrapassada pelo método *Branch and Price*, proposto por Novak et al. (2022). O método *Branch and Price* foi implementado em C++ usando SCIP 6.0.0 , enquanto que os outros problemas formulados utilizando Programação Linear foram implementados em Python 3.7, usando o Gurobi. Foi estipulado um tempo de *timeout* para resolver cada problema. Caso falhe, essa não é incluída no tempo médio de processamento (Novak et al., 2022).

Um das desvantagens é a necessidade de tempo de processamento, que nem sempre está disponível quando se querer responder ás alterações do sistema em tempo real. Este tempo depende do número de tarefas a calendarizar e da dimensão dos problemas MILP, em termos de número de variáveis de decisão, restrições e até número de funções objetivo (Duan & Wang, 2022). Noutras situações, como a calendarização da produção para o período seguinte, o tempo de processamento não afeta o planeamento, pois pode ser feito durante a noite, em poucas horas (dependendo do poder de processamento dos equipamentos em questão) (Souza et al., 2022). Existem abordagens puramente reativas de planeamento onde o tempo de processamento não excede 1 segundo, o que em termos operacionais é uma mais valia para a reação atempada da calendarização a perturbações do sistema. Outro tempo importante a considerar será, neste caso, o tempo de *update* das bases de dados que guardam toda a informação recolhida ao longo do tempo sobre o estado do sistema (Manzini et al., 2022).

2.4 Comentários Finais

A implementação de AGVs e AMRs em unidades de fabrico e armazéns permite o aumento significativo da capacidade produtiva e da utilização eficaz e eficiente dos recursos humanos, materiais e energéticos que tem à disposição. Na Tabela 2.2, a presente revisão de literatura permitiu identificar quais os principais tipos de AGV e AMR, quais os desafios na sua implementação, por fim, como é que a Calendarização Robusta pode ser utilizada como ferramenta de alocação de tarefas a recursos diversos.

Tabela 2.2: Resposta às questões de investigação da Revisão Sistemática

Questões de Investigação	Respostas
Q1: Quais os principais tipos de sistema de navegação e controlo de AGVs and AMRs?	**Sistema de Navegação:** Bandas magnéticas; Leitores de código de barras; RFID; Laser; Sistemas de navegação ótico, ultrassónico ou visuais; **Sistemas de controlo:** Clássico; Inteligente; Robusto; Preditivo; Ótimo; Adaptativo
Q2: Quais os desafios encontrados durante a implementação destes sistemas?	Algoritmos de Calendarização de Veículos e de Definição de Rotas; Dependência energética; Segurança de veículos, operadores e outros equipamentos; Resistência de Operadores e Gestão durante a implementação;
Q3: Como é avaliada a performance destes sistemas em ambiente fabril?	Tempos de reposição do material; Pontualidade do veículo; Eficiência na descarga; Número de tarefas; Número de encomendas satisfeitas; Tempo médio de execução de uma encomenda; Número de incidentes que o veículo incorre por dia; Custo por viagem ou transporte;
Q4: Como é que MILP pode ser usada para a formulação de problemas de RS?	Para formular um problema de Programação Linear Inteira Mista, no contexto da RS, é necessário: • Definir o problema como um problema de calendarização (PFSP, PMP, FJSP, entre outros); • Classificar e definir a Notação do problema, dependente da estrutura do problema, das características do ambiente produtivo e do tipo de planeamento desejado; • Escolher a Função Objetivo que melhor se adequa ao problema e que mais beneficia o decisor; • Identificar as Fontes de Incerteza Internas, Externas e Pontuais a considerar; • Escolher as Medidas de Desempenho da Robustez dos planeamento a construir; • Escolher os Algoritmos e/ou *Solvers* para resolver o problema MILP;

Calendarização Robusta enquanto ferramenta

A diversidade de problemas encontrados no que toda à calendarização de tarefas demonstra o quão flexível é a aplicação de métodos de Calendarização Robusta, com especial foco na

formulação MILP deste tipo de problemas.

A Programação Linear está entre os métodos mais utilizados, pela sua facilidade de interpretação e aplicação. O facto de várias funções objetivo e restrições serem transversais a problemas diferentes revela a adaptabilidade da ferramenta a diversos tipos de tarefas e recursos a calendarizar.

A analise detalhada revela ainda alguns aspetos a melhorar e novas frentes de investigação a explorar na área da Programação Linear e Calendarização Robusta no seu todo:

- Aplicação dos algoritmos MILP propostos a vários tipos de problemas e variações do mesmo tipo de problemas, de forma a avaliar o modelo segundo novas restrições (Cohen et al., 2023; Levorato et al., 2022);
- Incluir mais eventos discretos e dinâmicos que melhor representem a complexidade do ambiente, como adição da manutenção condicionada, com instante de ocorrência e durações estocásticas (Duan & Wang, 2022; Ghaleb & Taghipour, 2023; Souza et al., 2022);
- Utilizar Simulação de Eventos Discretos para aproximar o modelo proposto à realidade do sistema produtivo (Duan & Wang, 2022; Ghaleb & Taghipour, 2023)
- Adicionar o indicador *Overall Equipment Effectiveness* (OEE) como potencial indicador de desempenho não só do abastecimento através de veículos autónomos, mas também dos métodos de Calendarização Robusta mencionados (Pinto et al., 2019);
- Utilizar diferentes funções objetivo (a título de exemplo, o *Atraso Total, Total Flow Time* ou até combinações ponderadas destes), adicionar restrições (como incompatibilidade de máquina e tarefa) e perceber qual a influência da magnitude dos valores médios e variâncias dos *inputs* no tempo de resolução(Novak et al., 2022).

Capítulo 3

Estudo de Caso: Projeto AGiLE

Antes de introduzir a metodologia proposta, é fundamental começar por explicar o que é o projeto AGiLE, o cenário que está atualmente a ser implementado nas instalações da Volkswagen Autoeuropa, Lda. (AE), bem como o estudo anterior, no qual assenta a presente dissertação. Este último, realizado por Almeida (2022), dá resposta à necessidade do estudo da *performance* do abastecimento da linha de montagem da AE utilizando o sistema proposto pelo projeto AGiLE, bem como a sua comparação com uma abordagem hipotética onde é utilizada uma frota de *Automated Guided Vehicle* ou Veículos Automáticos Guiados (AGV)s para movimentar as *racks*. Como tal, obtou-se pela *Discreet Event Simulation* ou Simulação de Eventos Discretos (DES), através do software *Flexsim*, para observar o comportamento esperado do sistema e determinar qual o número de *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR) que melhor se adequa às necessidades da linha de montagem. Para este efeito, foram utilizados como *Key Performance Indicator* (KPI) a utilização média da frota de AMR, o número de peças mínimo, médio e máximo que, em média, se encontram em cada posto de trabalho, e ainda a distância média percorrida por cada robô.

3.1 Apresentação do Projeto AGiLE e Modelo Atual

O projeto AGiLE, liderado pela Imeguisa Portugal - Indústrias Metálicas Reunidas, S.A. e em co-promoção com a AE e o Instituto Superior Técnico, propõe uma solução descentralizada que garante a automatização *end-to-end* de processos intralogísticos numa linha de produção ou montagem, com especial atenção à automatização do transporte entre o Supermercado de manufatura (SUMA) e o *Point of Fit* (POF) ou local de montagem de três componentes distintos. Segundo Ohno e Bodek (2008), um Supermercado de Manufatura ou SUMA é um armazém intermédio onde estão armazenados os componentes ou *Work in Progress* (WIP) de processos-cliente e onde estes podem obter o que precisam, no momento certo e na quantidade certa. Esta automatização assenta na premissa de que existe uma comunicação constante entre os agentes de transporte e a linha de montagem, de forma a garantir que os componentes são fornecidos

de acordo com as necessidades da mesma. Desta forma, garante-se que os recursos disponíveis são utilizados da forma mais eficaz e eficiente possível (Imeguisa Portugal, 2021). Este projeto tem como objetivos:

- Desenvolver periféricos ou *racks* capazes de fornecer informação sobre a carga existente nos postos de trabalho e lançar pedidos de abastecimento, quando necessário;
- Capacitar os AMR para a gestão descentralizada de pedidos de abastecimento e otimização de rotas, através do planeamento coordenado de trajetórias entre todos os robôs;
- Criação de um sistema de monitorização capaz de registar e intervir no abastecimento para, por exemplo, reserva de agentes para manutenção preventiva ou indicação de zonas da fábrica encerradas temporariamente.

3.1.1 Descrição da arquitetura do sistema e características do abastecimento através do AGiLE

O AGiLE é composto por três agentes principais: uma frota de três AMR, periféricos sensorizados e uma entidade supervisora artificial. Estes estão ligados por uma estrutura em rede que elimina a possibilidade de um *single point of failure*. Um Single Point of Failure é uma entidade na arquitetura de um sistema que, quando falha, compromete o correto funcionamento de todas as outras entidades e, consequentemente, de todo o sistema. Um exemplo seria uma falha de Internet em todos os computadores devido ao mau funcionamento do *router*. A arquitetura atual do sistema procura ter a escalabilidade e flexibilidade necessária, não só para uma futura implementação na AE, mas também como um futuro produto a comercializar (Imeguisa Portugal, 2021).

Todos os AMR têm a capacidade de navegar livremente e evitar obstáculos estáticos e dinâmicos num ambiente de produção, bem como de comunicar com os periféricos e com a entidade supervisora. Já os periféricos ou *racks* podem ser divididos em duas categorias:

- ***Racks* de Bordo de Linha (r_{BL})** - Estruturas fixas na linha de montagem e onde os componentes são colocados após o transporte e são geralmente utilizadas para componentes não sequenciados
- ***Racks* de Transporte (r_T)** - Estruturas móveis que transportam componentes sequenciados do SUMA para o POF. Quando estas *racks* se encontram nos respetivos POF, funcionam também como *racks* de bordo de linha, de onde são retirados os componentes prontos a montar;

De acordo com a filosofia *Just-in-Time*, um componente sequenciado partilha um código único com o produto final onde vai ser montado e é entregue pelo fornecedor no momento da montagem. A sequenciação na *rack* de transporte pode ser feita durante o processo de *picking* (no SUMA) ou pelo próprio fornecedor, antes da entrega.

Para estudar o abastecimento da linha de montagem através da abordagem proposta, são utilizados três processos (representados na Figura 3.1, associados a três componentes diferentes):

- Aro da Cava da Roda do lado esquerdo (ACR-LH) (representado pelo índice 1);

- Tampas e Filtros (TF) (representado pelo índice 2);
- Módulos do Motor (MM) (representado pelo índice 3).

Cada um destes componentes tem *racks* alocadas ao transporte dos mesmos. O componente ACR-LH tem à disposição **3 *racks*** de transporte, com capacidade para **24 peças**. O mesmo acontece com o componente TF, que tem alocadas **2 *racks*** com capacidade para **18 peças**. Já o componente MM tem apenas **1 *rack*** de transporte de caixas de **12 peças cada**, sendo que cada *rack* transporta 2 caixas (ou seja, **24 peças**). Esta única *rack* de transporte, quando movimentada por um AMR até ao local de descarga, descarrega apenas as caixas (por um mecanismo denominado de *shooter*, onde as caixas cheias ou vazias são carregadas nas *racks* de bordo de linha ou no SUMA, fixas no local).

Foi ainda proposto um quarto processo: o Aro da Cava da Roda do lado direito (ACR-RH) (representado pelo índice 4 na Figura 3.1), como *backup* e, principalmente, como um ponto de fácil expansão para aumentar a complexidade do sistema (Imeguisa Portugal, 2022). Considera-se que este componente utiliza **3 *racks*** com capacidade para **24 peças**, iguais às do componente ACR-LH.

Os componentes têm SUMA e POF diferentes, cuja localização é apresentada na Figura 3.1, considerando os 4 processos referidos através dos índices 1, 2, 3, 4, e onde:

- Triângulo - Localização dos SUMA S_1, S_2, S_3 e S_4;
- Círculo - Localização dos POF P_1, P_2, P_3 e P_4;
- Círculo Duplo - Localização do processo intermédio Pré-Montagem (PM), necessário ao componente MM. É realizado antes da montagem dos MM no POF P_3.

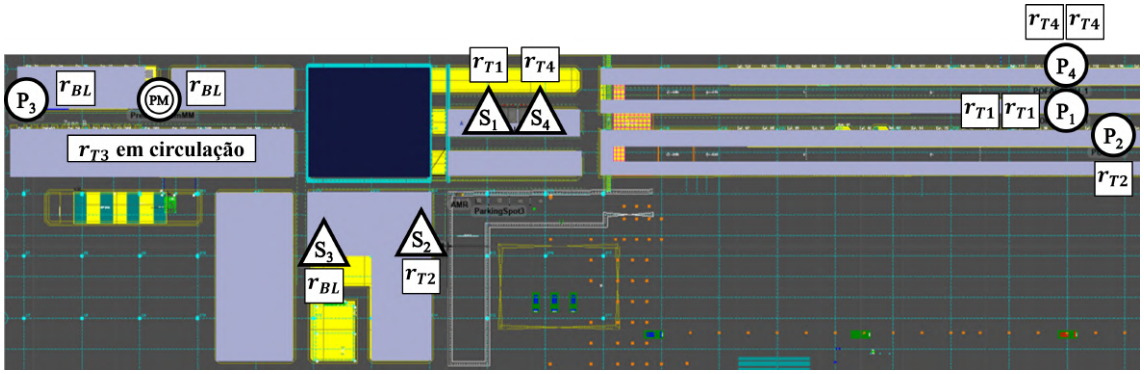


Figura 3.1: *Layout* da linha de montagem final na AE, no qual é representado cada SUMA e POF. Os planos cinzentos e azul escuro representam áreas onde é interdita a entrada de AMRs.

O abastecimento dos quatro componentes implica a movimentação de *racks* cheias dos SUMA para os POF respetivos, bem como a movimentação de *racks* vazias dos POF para os SUMA. As movimentações são iniciadas de acordo com eventos *trigger*, que despoletam a emissão de um pedido de movimentação de uma *rack* (Tabela 3.1). Representam geralmente o término/início de um processo ou o número de peças ainda por consumir numa *rack*. O componente ACR-RH, se incluído no abastecimento pelo AGiLE, tem *triggers* iguais aos do componente ACR-LH, dado que se trata do mesmo componente, montado em lados opostos da linha.

Quando os *triggers* associados a determinado pedido de movimentação são acionados, o pedido é recebido pela frota de AMRs, que determina qual o AMR livre mais próximo da *rack* a movimentar. Esta é a abordagem implementada no projeto e que se pretende comparar com o modelo de Calendarização Robusta proposto.

Tabela 3.1: Eventos *trigger* para emissão de um pedido de movimentação por componente. Adaptado de (Almeida, 2022).

Ponto de Recolha	ACR-LH	TF	MM
SUMA	*Rack* sai do POF (posição livre no POF) e *Picking* está concluído	*Picking* está concluído e *Rack* sai do POF (posição livre no POF)*	**W=12** peças por consumir no local da pré-montagem
POF	**X=4** Peças no *rack* que está a ser consumido no POF	**Y=4** peças na *rack* que está a ser consumido no POF	Descarga de caixas cheias está concluída e Carga de caixas vazias está concluída
Pré-Montagem	-	-	**Z=12** peças por consumir no POF e *Rack* cheio na Pré-Montagem

*Este *trigger* é redundante, pois no POF existem duas posições para as *racks* de transporte do componente TF , existindo sempre uma posição livre no POF quando uma das *racks* está no SUMA.

Atualmente, a combinação do número de peças **W**,**X**,**Y** e **Z** (Tabela 3.1) dá origem aos pedidos de movimentação de *racks*. Esta combinação é usada no estudo sobre o abastecimento da linha de montagem realizado (Almeida, 2022), durante a implementação do AGiLE na AE, e foi definida pelos responsáveis do projeto.

O momento em que estes *triggers* são ativados depende de vários fatores, detalhados por Almeida (2022), tais como o tempo de *picking* de cada componente nos respetivos SUMA, tempo de carga e descarga dos robôs no inicio e fim de uma tarefa de abastecimento (tempo de *load* e *unload*), velocidade dos AMR, tempo de *load* e *unload* de caixas no processo de abastecimento do componente MM, para além do número de peças que indica a necessidade de peças junto à linha de montagem.

O processo de *picking* é definido como o processo de recolha de componentes ou WIP em armazéns e supermercados, de forma a responder ao pedido de um processo ou cliente específico (De Koster et al., 2007). O tempo de *load* e *unload* de um robô corresponde ao tempo decorrido deste o momento em que um robô para junto local de carga ou descarga de uma *rack*, carrega ou descarrega a *rack* nesse local e inicia novamente a marcha.

Na secção seguinte, são descritas algumas alterações ao modelo 3D da linha de montagem final. Segue-se depois a análise dos tempos de *picking* de cada componente (fornecidos pela AE), bem como os tempos de *load* e *unload* dos AMR medidos.

3.2 Sugestões de Alteração do Modelo Atual

Com vista ao desenvolvimento do método de Calendarização Robusta proposto, importa analisar previamente o modelo de simulação, para que este se aproxime dos objetivos e especificações determinados no início do projeto, bem como fornecer mais informação sobre os agentes e processos intervenientes, nomeadamente nos processos de *picking* e carga e descarga de *racks* pelos AMR.

3.2.1 Alterações propostas do modelo 3D atual

A representação a três dimensões da linha de montagem é uma das componentes da DES com maior influência na performance do modelo, nas medidas de desempenho selecionadas e, como consequência, na tomada das decisões necessárias. Em chão de fábrica, a localização dos SUMA, POF ou Pré-Montagem é fixa e determinante para o cálculo da distância que um AMR tem que percorrer para desempenhar determinada tarefa. Assim, a representação virtual do sistema deverá estar de acordo com o *layout* da fábrica e com o número de *racks* que se podem colocar no local em questão. A modelação atual da linha de montagem da AE está representada na Figura A.1. É utilizado o software *Flexsim* para modelar o sistema atual e as seguintes propostas.

O modelo atual utiliza os objetos *queue* do *Flexsim* para assinalar as posições dos locais de recolha e entrega de *racks*. Estes objetos permitem depositar itens que representem componentes ou estruturas de transporte de componentes, como as *racks* já mencionadas. Nos componentes ACR-LH e TF, em particular, o modelo utilizava uma *queue* para cada *rack*, o que não acontece na realidade, de acordo com as especificações do projeto (Imeguisa Portugal, 2022). Para além destes dois componentes, existe ainda o componente ACR-RH, que pode ser incorporado no mesmo modelo, após as alterações necessárias, e cujas posições das *racks* precisam de ser marcadas por novas *queues*.

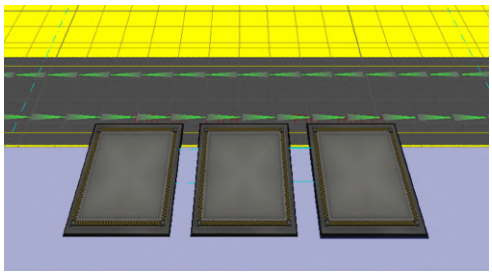
Os componentes ACR-LH e ACR-RH partilham o espaço no SUMA, onde são necessárias quatro *queues* (duas para cada componente) (Figura 3.2). As três *queues* usadas no modelo atual, exclusivas ao componente ACR-LH (Figura 3.2a), são substituídas por quatro *queues* que marcam a posição das *racks* de cada um dos componentes (Figura 3.2b).

No POF do componente ACR-LH (Figura 3.3) existem três *queues* (Figura 3.3a) que são substituídas por duas *queues* (Figura 3.3b), correspondentes aos dois locais onde podem ser colocadas *racks* cheias junto à linha de montagem.

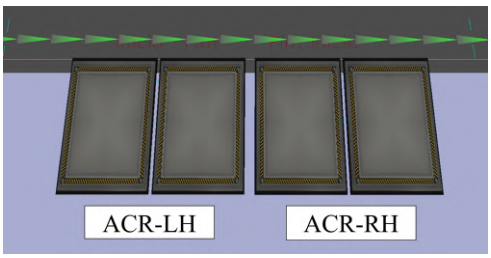
No POF do componente ACR-RH (Figura 3.4), tal como no POF do componente ACR-LH, existem duas posições bem definidas para a colocação de *racks* junto à linha. A sua localização está marcada no *layout* da linha de montagem (Figura 3.4a). Daqui em diante, as duas racks podem ser colocadas numa das posições representadas na Figura 3.4b.

No POF do componente TF, as duas *queues* mantém-se, visto que estão de acordo com as duas posições para *racks*, junto à linha de montagem. Já no SUMA (Figura 3.5), as duas *queues* usadas (Figura 3.5a) são substituídas por uma *queue* (Figura 3.5b).

Desta alteração resulta a necessidade de, no inicio de cada transporte, informar o AMR sobre qual a posição disponível no destino, para que possa deslocar-se e entrar uma *rack* num lugar vazio. Os triggers correspondentes a tarefas de movimentação de *racks* cheias do SUMA

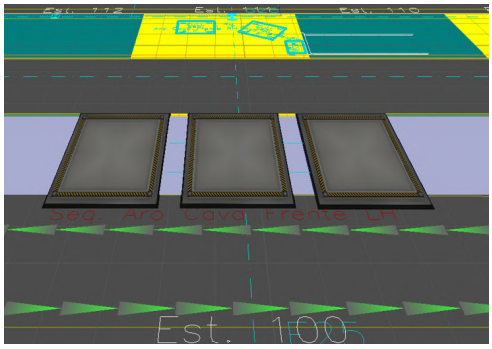


(a) Modelação atual, sem as *queues* correspondentes ao componente ACR-RH

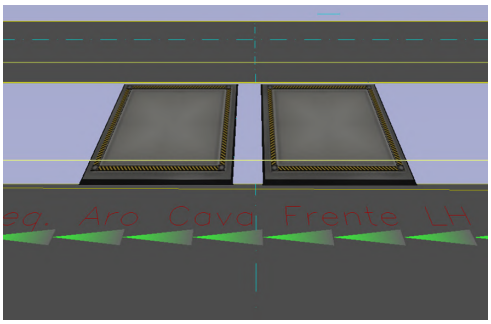


(b) Modelação proposta

Figura 3.2: *Queues* no SUMA dos componentes ACR-LH e ACR-RH

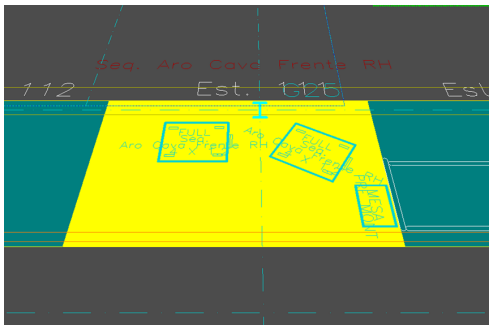


(a) Modelação atual

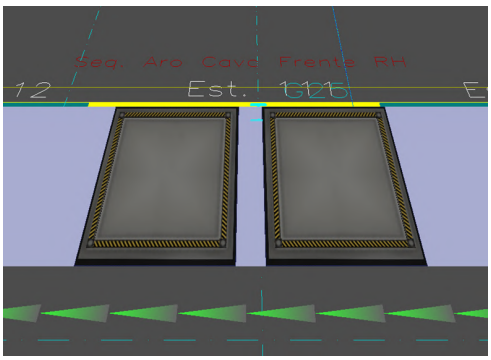


(b) Modelação proposta

Figura 3.3: *Queues* no POF do componente ACR-LH



(a) Modelação atual, sem nenhuma *queue*, por corresponder a um componente que não foi simulado no modelo atual

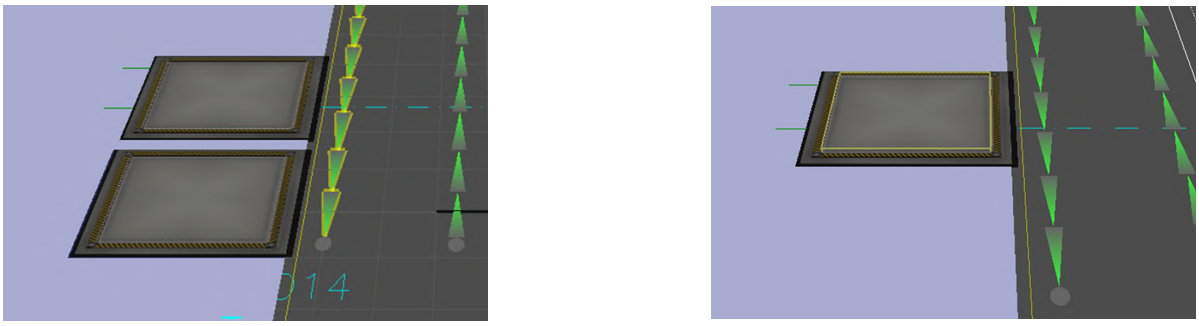


(b) Modelação proposta

Figura 3.4: *Queues* no POF do componente ACR-RH

para o POF já garantem que existe uma posição vazia no mesmo. No entanto, as tarefas de movimentação de *racks* vazias dos POF para os SUMA correspondentes não têm essa garantia. Dado que o tempo de *picking* de uma *rack*, para os componentes ACR-LH e TF, é inferior ao tempo de consumo de uma *rack* no POF (1728 e 1296 segundos, respetivamente), os processos de entrega de *rack* vazia no SUMA, *picking* e entrega de *rack* cheia no POF ocorrem antes da chegada de uma nova *rack* vazia ao SUMA. Isto é principalmente importante para o componente TF, visto que só tem uma posição disponível no SUMA. Após alteração do número e localização de *queues* no modelo de simulação, a Figura A.2 no Apêndice A representa o *layout* do modelo a utilizar durante o desenvolvimento do método de Calendarização Robusta.

]



(a) Modelação atual

(b) Modelação proposta

Figura 3.5: *Queues* no SUMA do componente TF

3.2.2 Análise dos Tempos de *Picking* e de *Load/Unload*

Todos os sistemas reais incluem fontes de aleatoriedade que afetam a validade e credibilidade dos modelos matemáticos e de simulação construídos. Embora seja impossível descrever por completo o comportamento de um sistema real (dado o elevado número de fontes de aleatoriedade), a quantidade e qualidade das distribuições probabilísticas de *input* contribuem para a menor ou maior complexidade do modelo desenvolvido. Em modelos de simulação, a informação de *input* a utilizar pode ser introduzida no modelo de três formas diferentes (Law, 2013):

- *Trace-driven Simulation* - Dados reais são diretamente utilizados como *input* da simulação;
- Distribuição Empírica - Geração de uma amostra, considerando a frequência relativa de intervalos contínuos dos valores observados e aproximando a frequência relativa por uma função de densidade empírica;
- Inferência Estatística - Ajustamento de uma distribuição de probabilidade teórica e dos seus parâmetros aos dados reais, introduzida posteriormente no modelo de simulação;

Neste caso, será feito o ajustamento de uma distribuição de probabilidade teórica que melhor se ajuste aos tempos de *picking* e de *load/unload* dos AMRs. No estudo anterior (Almeida, 2022), foram igualmente estudados os tempos de *picking* através do software ExpertFit. De forma a garantir que são escolhidas as distribuições mais ajustadas de *input* do modelo, os tempos de *picking* serão novamente analisados, juntamente com os dados recolhidos sobre os tempos de *load/unload* dos robôs. Para o efeito, seguiram-se os seguintes passos enumerados na Figura 3.6, implementados em *Python*.

Na Etapa 1 (Figura 3.6) é utilizado um gráfico *boxplot* para identificar e eliminar elementos de cada amostra como *outliers*. O método *boxplot* é um método gráfico simples introduzido por Tukey (1977) para representar elementos de uma amostra, relacionando-os com a dispersão, mediana e limites inferior (*LI*) e superior (*LS*) da respetiva amostra (Sim et al., 2005). Em particular, os *LI* e *LS* são definidos pelo utilizador, dependendo do tipo de dados que se pretende analisar. Dada a dimensão das amostras (superior a 400 elementos) e ao facto das distribuições que melhor se ajustam serem desconhecidas, os *LI* e *LS* são calculados através das Equações (3.1), (3.2) e (3.3):

$$\text{IQR} = q_3 - q_1 \tag{3.1}$$

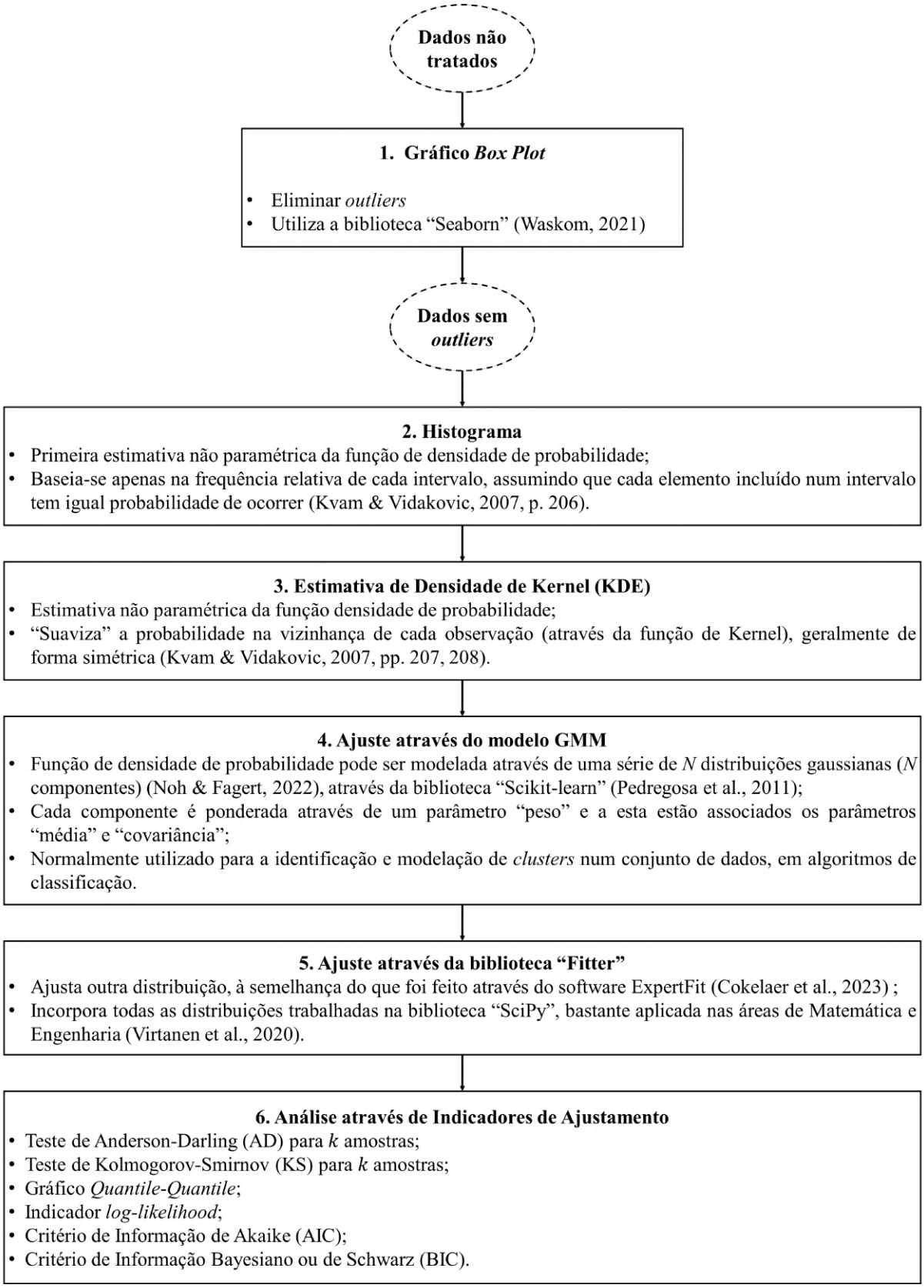


Figura 3.6: Etapas da análise dos tempos de *picking* e tempos de *load/unload* recolhidos.

$$\text{LI} = \text{q}_1 - 1.5\text{IQR} \tag{3.2}$$

$$\text{LS} = \text{q}_3 + 1.5\text{IQR}, \tag{3.3}$$

onde q_1 é o percentil 25, q_3 é o percentil 75 e IRQ é o intervalo interquartil. Embora estes limites sejam mais adequados para distribuições simétricas (característica desconhecida em relação aos dados a analisar), é um método simples para identificar elementos mais extremos em cada amostra e que pareçam, à partida, inconsistentes com o comportamento observado nos restantes elementos da amostra. Todos os elementos da amostra que se encontram abaixo do LI ou acima do LS são retirados da amostra.

Na Etapa 2 da Figura 3.6, é utilizada a *Kernel Density Estimation* ou Estimativa de Densidade de Kernel (KDE) para obter uma estimativa não paramétrica da função de densidade de probabilidade ($\hat{f}(x)$) que melhor se ajusta à amostra em estudo, através da biblioteca "Seaborn" (Waskom, 2021). Esta estimativa é calculada pela Equação 3.4:

$$\hat{f}(x) = \frac{1}{nh_n} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h_n}\right), \tag{3.4}$$

onde n é a dimensão da amostra, x é uma incógnita, x_i é o i-ésimo valor da amostra e h_n é a largura de banda da estimativa ou parâmetro de suavização da estimativa (conceito semelhante aos intervalos de um histograma). A Função de Kernel $K(x)$ determina a forma como a "suavização" dessa probabilidade é feita. Segundo a documentação da biblioteca "Seaborn", a função $K(x)$ utilizada é a Função Normal ou Gaussiana, que é a mais usual. Isto significa que a cada x_i corresponde uma função de densidade de probabilidade Normal (chamada *Kernel*) centrada em $\mu = x_i$ e com desvio padrão $\sigma = h_n$. A soma dos n *Kernels* gaussianos, multiplicados pelo fator $\frac{1}{nh_n}$, fornecem uma estimativa da função densidade de probabilidade da amostra em estudo (Chen, 2017).

Nas Etapas 4 e 5 (Figura 3.6), serão ajustadas distribuições contidas nas bibliotecas "Scikit-learn" e "Fitter", de forma a ajustar cada amostra a uma mistura de N distribuições gaussianas (*Gaussian Mixture Model* ou Modelo de Mistura Gaussiana (GMM)) e às distribuições incluídas na biblioteca "SciPy", respetivamente. O Modelo de Mistura Gaussiana é uma função densidade de probabilidade que corresponde à soma ponderada de N distribuições gaussianas. O GMM permite categorizar elementos de uma amostra, identificando sub-amostras que sejam melhor ajustadas por funções densidade de probabilidade gaussianas diferentes. Assim, a amostra poderá ser descrita pela soma ponderada das distribuições gaussianas que caracterizam cada sub-amostra que a constitui (Wan et al., 2019). Esta característica permite utilizar o GMM em problemas de classificação e identificar características dos dados em estudo que estejam, aparentemente, escondidas. Dado que o GMM será utilizado para ajustar uma distribuição aos tempos de *picking* de operadores nos SUMA e de *load/unload* dos robôs, podemos utilizar este modelo para, futuramente, identificar padrões relacionados com a performance dos operadores (que trabalham por equipas em 3 turnos diferentes ao longo do dia) e dos robôs, respetivamente.

São utilizadas bibliotecas de suporte à análise de dados, como as bibliotecas "Pandas" (McKinney, 2010) e "Numpy" (Harris et al., 2020).

O Teste de *Anderson-Darling* (AD) e Teste de *Kolmogorov-Smirnov* (KS), para k amostras, testam se duas ou mais amostras independentes foram retiradas da mesma população com determinado nível de confiança, sem, à partida, especificar uma distribuição que descreve essa mesma população. Para isto, são geradas amostras das distribuições ajustadas nas Etapas 4 e 5, que, por sua vez, são comparadas individualmente com os dados reais sem *outliers*. Para ambos os testes, a hipótese nula (H_0) representa o caso em que as amostras pertencem à mesma população, ou seja, podem ser descritas pela mesma distribuição de probabilidade. Em contrapartida, a hipótese H_1 representa o caso em que pelo menos uma amostra é descrita por uma distribuição diferente. Considera-se que o Teste de *Anderson-Darling* para k amostras é mais robusto do que o Teste de *Kolmogorov-Smirnov* para k amostras, em particular quando são testadas distribuições assimétricas, pois o AD foca-se na cauda das distribuições (Razali & Wah, 2011). Isto é importante, dado que os valores extremos, localizados nas caudas da distribuição, podem afetar significativamente a performance de um modelo de simulação, embora a probabilidade de ocorrência seja mínima. O Teste de *Kolmogorov-Smirnov* para k amostras, embora não seja o mais indicado para testar o ajuste na cauda das distribuições, fornece alguma informação sobre a localização da distribuição face à amostra, pois é mais sensível perto do centro das distribuição (Bishop, 2006). Assim, a conjugação dos testes AD e KS permite tirar conclusões não só sobre o ajuste ao centro da distribuição como também nas periferias da amostra. Ambos os testes são aplicados através da biblioteca "Scipy" (Virtanen et al., 2020).

Sendo o nível de confiança $(1 - \alpha)$ de 0.95 (95%), a hipótese H_0 será rejeitada quando o nível de significância $\alpha = 0.05$ é superior ao $p - value$ relativo ao valor da estatística de teste. Segundo a documentação do teste *Anderson-Darling* na biblioteca "Scipy" (Virtanen et al., 2020), o resultado $p - value \geq 0.25$ indica a existência de evidência estatística para concluir que não se deve rejeitar a H_0, mas sem conseguir especificar o $p - value$ verdadeiro. Neste caso, é usada a estatística de teste associada para a tomada de decisão.

O gráfico *Quantile-Quantile* (*Q-Q plot*) estabelece uma comparação gráfica entre os quantis de diferentes distribuições de probabilidade. A partir dos dados observados reais e da amostra da distribuição a analisar (ordenados por ordem crescente), são calculados níveis homogéneos para cada conjunto de dados. De seguida, é feita uma interpolação linear através dos níveis calculados da amostra e dos respetivos valores gerados. Essa interpolação permite calcular o valor estimado do quantil da distribuição, quando fornecido um nível calculado a partir dos dados observados reais. Para cada distribuição a analisar é feita uma interpolação linear e calculados os valores estimados dos quantis relativamente aos valores reais. É utilizada uma reta de referência associada aos dados reais para comparar com os valores estimados para cada distribuição. Se a representação dos dados se afastar muito da reta de referência, então o ajustamento não é adequado.

O ajuste da amostra de valores reais $X = \{x_1, ..., x_n\}$ a uma distribuição com função de densidade de probabilidade f (através do modelo GMM e da biblioteca "Fitter") implica a existência de um vetor de parâmetros $\boldsymbol{\theta}$, com dimensão k, que melhor ajusta f aos dados reais (Konishi & Kitagawa, 2008). O indicador *log-likelihood* ($l(\boldsymbol{\theta_{MLE}})$) pretende medir a qualidade desse mesmo ajuste. É utilizada a Função de Verosimilhança ($L(\boldsymbol{\theta})$), cujo valor é calculado através da Equação 3.5:

$$\mathrm{L}(\boldsymbol{\theta}) = \prod_{\mathrm{i}}^{\mathrm{n}} \mathrm{f}(\mathrm{x_i}|\boldsymbol{\theta}), \tag{3.5}$$

e onde i é o índice de cada elemento da amostra X, com n elementos, e $\boldsymbol{\theta}$ é o vetor de parâmetros da distribuição de densidade de probabilidade f.

Dado que a função logaritmo é estritamente crescente e que o objetivo é maximizar a Função de Verosimilhança, $l(\boldsymbol{\theta_{MLE}})$ corresponde ao logaritmo da Função de Verosimilhança quando $\boldsymbol{\theta}$ é o vetor de parâmetros que maximiza $L(\boldsymbol{\theta})$, ou seja, assumindo que $\boldsymbol{\theta}$ é o Estimador de Máxima Verosimilhança ($\boldsymbol{\theta_{MLE}}$). Desta forma, a distribuição que melhor se ajusta aos dados reais é aquela cujo $l(\boldsymbol{\theta_{MLE}})$ do ajuste é maior (Konishi & Kitagawa, 2008).

Os critérios *Bayesian or Schwarz Information Criterion* ou Critério de Informação Bayesiano ou de Schwarz (BIC) e *Akaike Information Criterion* ou Critério de Informação de Akaike (AIC) são critérios de seleção de modelos de distribuição de probabilidade, baseados no indicador $l(\boldsymbol{\theta_{MLE}})$ e que pretendem avaliar o quão próximo é o modelo estatístico ajustado da distribuição de probabilidade real. Chakrabarti e Ghosh (2011), ao comparar ambos os critérios, percebeu que o critério BIC é mais útil para garantir que estamos a escolher a distribuição que melhor se ajusta à amostra. O critério AIC, por sua vez, permite escolher a distribuição que resulta numa melhor previsão de observações futuras. Dado que o objetivo é utilizar a distribuição ajustada para, no futuro, prever tempos de *picking* e de *load/unload* dos robôs em modelos de simulação, a utilização do critério BIC em conjunto com o critério AIC é uma mais valia.

O critério BIC (Equação 3.6) é calculado a partir da probabilidade marginal $p(x_n)$) e das probabilidades *a priori* e *a posteriori* do modelo estatístico ajustado, assumindo que a dimensão da amostra (n) é elevada (Konishi & Kitagawa, 2008, p. 217).

A probabilidade marginal é definida como a distribuição de probabilidade de uma variável aleatória do modelo, independentemente de outras variáveis aleatórias que possam fazer parte do modelo estatístico (Bishop, 2006, p. 13). O valor desta medida é dado por:

$$\mathrm{BIC} \simeq -2\mathrm{l}(\boldsymbol{\theta}) + \mathrm{k}\log\mathrm{n} \tag{3.6}$$

A probabilidade *a priori* de um modelo estatístico é a probabilidade de um modelo ser aquele que melhor se ajusta à distribuição real(Konishi & Kitagawa, 2008, p. 211). A probabilidade *a posteriori* de um modelo estatístico é a probabilidade condicionada de um modelo ser aquele que melhor se ajusta à distribuição real, sabendo que foram observados os elementos de uma amostra x_n (Konishi & Kitagawa, 2008, p. 212).

O critério AIC (Equação 3.7), por outro lado, ajusta o indicador $l(\boldsymbol{\theta_{MLE}})$ através do número de parâmetros da distribuição ajustada. Para o efeito, parte-se do pressuposto que os parâmetros estimados pertencem a um modelo paramétrico que já contém o valor real dos parâmetros da distribuição real (Konishi & Kitagawa, 2008, p. 60).

$$\mathrm{AIC} = -2\mathrm{l}(\boldsymbol{\theta}) + 2\mathrm{k} \tag{3.7}$$

Em ambos os critérios, o vetor $\boldsymbol{\theta}$ corresponde ao vetor de parâmetros estimados do modelo estatístico ajustado que maximiza o logaritmo da Função de Verosimilhança ($\boldsymbol{\theta_{MLE}}$) e k corresponde ao número de parâmetros estimados da distribuição de probabilidade ajustada.

Como o BIC e o AIC dependem do $l(\boldsymbol{\theta}_{MLE})$, quanto menor for o BIC, melhor será o ajustamento da distribuição aos dados originais. Da mesma forma, quanto menor o AIC, melhor será o ajustamento da distribuição aos dados originais. Tal como dito anteriormente, como não é feita a otimização de parâmetros para cada distribuição de probabilidade modelada, o vetor $\boldsymbol{\theta}$ é considerado como sendo o Estimador de Máxima Verosimilhança $\boldsymbol{\theta}_{MLE}$.

Após esta breve introdução aos testes e indicadores utilizados, é feita análise dos tempos de *Picking* e de *Load/Unload* dos AMR.

3.2.2.1 Tempos de *Picking*

Nesta secção serão novamente analisados os tempos de *picking*, fornecidos pela AE e referentes aos mês de maio de 2022. Através do teste não paramétrico de *Kruskal-Wallis*, o estudo anterior (Almeida, 2022) concluiu que os dados observados dos turnos de determinados componentes (amostras independentes) são estatisticamente diferentes. Este facto impediu a aglomeração das amostras dos turnos, decisão que é repetida no presente estudo. Após a análise de cada um dos turnos e não sabendo quais os valores de referência dos tempos de *picking*, é possível construir dois cenários possíveis para a escolha dos turnos que vão modelar o *picking* de cada um dos componentes: um cenário otimista e um cenário pessimista.

- O cenário pessimista pretende utilizar o turno com pior desempenho (maiores média e desvio padrão). Na realidade, este turno representa o pior caso possível no que toca ao abastecimento por parte dos robôs. Maiores médias e desvios padrão implicam uma maior probabilidade de um atraso no processo de *picking* provocar incumprimentos nos POF. Maiores desvios padrão, em particular, geram valores extremos e anormais com maior frequência, o que pode afetar negativamente a performance do abastecimento de *racks*.
- O cenário otimista pretende utilizar o turno com melhor desempenho (menores média e desvio padrão). Em contexto industrial, este é um turno cujo desempenho é exemplar, pressionando os restantes turnos do mesmo componente para melhorar a sua performance. Menores médias e desvios padrão garantem que as *racks* nos SUMA estão disponíveis mais rapidamente, diminuindo a probabilidade de um atraso no processo de *picking* provocar incumprimentos nos POF.

Após identificar e retirar os *outliers* de todos os turnos, através de gráficos *boxplot* (Figura B.1 à Figura B.16), é possível calcular a média e desvio padrão dos dados originais sem o impacto dos *outliers* retirados (Tabela B.1), Na Tabela B.1 pode observar-se também o tamanho da amostra de cada turno, antes e depois da eliminação dos *outliers*. Destes, podem identificar-se os turnos dos cenários pessimista e otimista, resumidos nas Tabelas 3.2 e 3.3, respetivamente. Quando a média e o desvio padrão de um turno não são ambos os maiores ou menores, o turno é escolhido tendo em conta também o valor mínimo e o valor máximo observado em cada turno.

No cenário pessimista, o turno MM_M3 é escolhido ao invés do turno MM_M4. O turno MM_M3 tem a maior média, mas não tem o maior desvio padrão. No entanto, é no turno MM_M3 que se observa o maior tempo máximo, o que indica a possibilidade de ocorrência de valores extremos com maior frequência, quando ajustada uma distribuição. Embora já tenham sido retirados os *outliers*, a maior variabilidade é indicativa da ocorrência de situações que podem

pôr em causa o abastecimento de *racks*, mesmo quando estão previstos na amostra recolhida já sem *outliers*.

Algo semelhante ocorre no cenário otimista dos componentes ACR-RH e TF. É escolhido o turno ACR_RH_M2, cuja média não é a menor, mas onde o desvio padrão é significativamente inferior ao dos restantes turnos e onde se observam os menores tempos mínimos e máximos da amostra. Para o turno do componente TF a escolha é mais difícil, pois o menor desvio padrão corresponde a um turno cujo média é a segunda mais alta. De acordo com a Figura B.17 e Tabela B.1, embora o turno TF_M1 corresponda a um maior desvio padrão, a conjugação com uma média menor permite que a ocorrência de tempos de *picking* superiores a 450 segundos seja menor (ao contrário do que acontece com o turno TF_M2). Esta decisão também pode ser confirmada pelos menores tempos mínimos e máximos observados na amostra do turno TF_M1.

Tabela 3.2: Tamanho da amostra, Média e desvio padrão dos tempos de *picking* dos turnos do cenário pessimista, após deteção e eliminação de *outliers*

Turnos do Cenário Pessimista					
Turnos	n_f	**Média (s)**	**Desvio padrão (s)**	**Mínimo (s)**	**Máximo (s)**
ACR_LH_M3	481	171.86	48.78	68	310
ACR_RH_M3	478	170.12	51.34	65	327
MM_M3	502	142.47	46.76	71	263
TF_M3	459	408.52	67.33	242	576

Tabela 3.3: Tamanho da amostra, Média e desvio padrão dos tempos de *picking* dos turnos do cenário otimista, após deteção e eliminação de *outliers*

Turnos do Cenário Otimista					
Turnos	n_f	**Média (s)**	**Desvio padrão (s)**	**Mínimo (s)**	**Máximo (s)**
ACR_LH_M2	515	127.19	29.70	61	210
ACR_RH_M2	515	124.15	29.11	61	209
MM_M1	414	125.69	22.65	70	187
TF_M1	513	336.14	65.75	150	520

De seguida, é construída uma estimativa não paramétrica da função densidade de probabilidade através da KDE (Figura B.18 à Figura B.33), procedendo-se também ao ajustamento dos dados reais através do GMM e da biblioteca "Fitter" (Figura B.34 à Figura B.49). Os resultados dos testes e indicadores de ajustamento estão descritos na Tabela B.2, acompanhados pelos gráficos *Quantile-Quantile* (Figura B.50 à Figura B.65). São apenas analisados os turnos selecionados em cada cenário, servindo os restantes resultados para posterior referência.

3.2.2.2 Turnos do Cenário Pessimista

O tempo de *picking* do turno ACR_LH_M3 pode ser modelado por uma **distribuição GMM com $N = 3$ componentes**, dado que:

- No Histograma e KDE da Figura B.20 observam-se pelo menos 3 picos, indicativos de, pelo menos, $N = 3$ componentes do modelo GMM;

- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.36 e B.52, respetivamente) confirmam que tanto o modelo GMM com $N = 3$ e $N = 4$ componentes é adequado, principalmente quando comparadas as caudas das distribuições ajustadas com a reta de referência.

- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;

- A estatística do teste AD é a segunda mais baixa, o que indica ter o segundo $p - value$ mais elevado. O mesmo acontece com o teste KS, onde $p - value$ é o segundo mais alto (muito próximo do $p - value$ da distribuição do modelo GMM com $N = 4$ componentes). A grande diferença entre as estatísticas de teste da GMM com $N = 3$ componentes e da GMM com $N = 2$ componentes justifica o aumento da complexidade;

- Os indicadores $l(\boldsymbol{\theta_{MLE}})$ e AIC das distribuições do modelo GMM com $N = 3$ e $N = 4$ componentes são próximos;

- O indicador BIC da distribuição GMM com $N = 3$ componentes é menor do que o BIC da distribuição GMM com $N = 4$ componentes;

O tempo de *picking* do turno ACR_RH_M3 pode ser modelado por uma **distribuição GMM com $N = 3$ componentes**, pois:

- No Histograma e KDE da Figura B.24 observam-se pelo menos 3 picos, indicativos de, pelo menos, $N = 3$ componentes do modelo GMM;

- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.40 e B.56, respetivamente) confirmam que a distribuição GMM com $N = 3$ componentes é suficiente para garantir um bom ajustamento aos dados reais, principalmente nas caudas da distribuição;

- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;

- Indicadores $l(\boldsymbol{\theta_{MLE}})$ entre os mais altos e indicadores BIC e AIC são os mais baixos entre as distribuições cujo $p - value \geq 0.25$ do teste AD;

- A distribuição GMM com $N = 3$ componentes é a distribuição mais simples cuja estatística do teste AD é mais baixa e cujo $p - value$ do teste KS está entre os mais altos;

O tempo de *picking* do turno MM_M3 pode ser modelado por uma **distribuição GMM com $N = 3$ componentes**, pois:

- No Histograma e KDE da Figura B.28 observam-se pelo menos 2 picos, indicativos de, pelo menos, $N = 2$ componentes do modelo GMM;

- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.44 e B.60, respetivamente) confirmam que a distribuição GMM com $N = 3$ é a distribuição menos complexa e que melhor se ajusta aos dados reais, tanto nas extremidades da distribuição como nos valores centrais;
- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- A distribuição GMM com $N = 3$ componentes é a distribuição com menor estatística do teste AD (maior $p - value$ esperado) e com $p - value$ do teste KS mais alto;
- Os indicadores BIC e AIC estão entre os mais baixos, entre as distribuições GMM com mais componentes;

O tempo de *picking* do turno TF_M3 pode ser modelado por uma **distribuição GMM com $N = 3$ componentes**, dado que:

- No Histograma e KDE da Figura B.32 observam-se pelo menos 3 picos, indicativos de, pelo menos, $N = 3$ componentes do modelo GMM;
- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.48 e B.64, respetivamente) confirmam que a distribuição GMM com $N = 3$ é a distribuição menos complexa e que melhor se ajusta aos dados reais, tanto nas extremidades da distribuição como nos valores centrais;
- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- A distribuição GMM com $N = 3$ componentes é a distribuição mais simples com menor estatística do teste AD (maior $p - value$ esperado) e com $p - value$ do teste KS mais alto. A grande diferença entre as estatísticas de teste desta distribuição e a GMM com $N = 2$ componentes justifica o aumento da complexidade;
- Os indicadores BIC e AIC estão entre os mais baixos, entre as distribuições GMM com mais componentes;

Após a escolha das distribuições que modelam os tempos de *picking* de cada turno do Cenário Pessimista, a Tabela 3.4 apresenta os parâmetros associados a cada distribuição.

Tabela 3.4: Parâmetros das distribuições que modelam os tempos de *picking* dos turnos do cenário pessimista. O parâmetro **Covariância** é gerado através do atributo $covariances\_ =$ *"full"*, depois do ajustamento aos dados reais. Os parâmetros não são arredondados para preservar o ajustamento feito.

Turnos do Cenário Pessimista				
		Parâmetros		
Turno	**Distribuição**	**Média**	**Covariância**	**Peso**
ACR_LH_M3	GMM ($N = 3$)			
	Componente 1	176.18676406	542.80263197	0.45617428987788444
	Componente 2	115.58848235	346.31849916	0.2880992459196901
	Componente 3	227.54766294	1248.2429921	0.2557264642024254
ACR_RH_M3	GMM ($N = 3$)			
	Componente 1	230.67081514	1238.54864732	0.22692923457054268
	Componente 2	118.41187702	461.69961319	0.38549123753004405
	Componente 3	186.08405512	554.18665395	0.3875795278994132
MM_M3	GMM ($N = 3$)			
	Componente 1	171.27390044	364.59406804	0.45667395940381417
	Componente 2	135.18887284	502.71971465	0.36035918893135355
	Componente 3	223.56224332	569.38492778	0.18296685166483234
TF_M3	GMM ($N = 3$)			
	Componente 1	497.33856974	1542.89418966	0.2204229780400245
	Componente 2	408.15940224	958.81597502	0.5213176895960752
	Componente 3	333.42754309	1928.54502135	0.2582593323639004

3.2.2.3 Turnos do Cenário Otimista

O tempo de *picking* do turno ACR_LH_M2 pode ser modelado por uma **distribuição GMM com $N = 2$ componentes**, dado que:

- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.35 e B.51, respetivamente) demonstram que a distribuição do modelo GMM com $N = 2$ componentes corresponde a um bom ajuste em ambas as caudas;
- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- As estatísticas dos testes AD e KS do modelo GMM com $N = 5$ componentes correspondem ambas ao $p - value$ mais elevado, contrariando a tendência de escolha da distribuição mais simples que melhor se ajusta;
- O indicador $l(\boldsymbol{\theta_{MLE}})$ não é um dos menores. No entanto, tanto o indicador BIC como o indicador AIC são os mais baixos;

O tempo de *picking* do turno ACR_RH_M2 pode ser modelado por uma **distribuição GMM com $N = 3$ componentes**, dado que:

- No Histograma e KDE da Figura B.23 observam-se pelo menos 2 picos, indicativos de, pelo menos, $N = 2$ componentes do modelo GMM;
- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.39 e B.55, respetivamente) demonstram que a distribuição do modelo GMM com $N = 3$ componentes é a distribuição mais simples que melhor se ajusta aos dados reais;
- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- As estatísticas dos testes AD e KS do modelo GMM com $N = 3$ componentes correspondem ambas ao terceiro $p - value$ mais elevado. No entanto, os indicadores BIC e AIC são os menores das 3 distribuições com maior $p - value$, o que suportam a escolha do modelo GMM com $N = 3$ componentes pela sua simplicidade e bom ajuste;

O tempo de *picking* do turno MM_M1 pode ser modelado por uma **distribuição GMM com $N = 2$ componentes**, dado que:

- No Histograma e KDE da Figura B.26 observam-se pelo menos 2 picos, indicativos de, pelo menos, $N = 2$ componentes do modelo GMM;
- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.42 e B.58, respetivamente) demonstram que a distribuição do modelo GMM com $N = 2$ componentes é a distribuição mais simples que melhor se ajusta aos dados reais;
- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- Embora as estatísticas dos testes AD e KS do modelo GMM com $N = 2$ componentes correspondam aos terceiros $p - values$ mais elevados e um indicador $l(\boldsymbol{\theta}_{MLE})$ menor, esta é uma distribuição mais simples, com menores indicadores BIC e AIC;

O tempo de *picking* do turno TF_M1 pode ser modelado por uma **distribuição GMM com $N = 2$ componentes**, dado que:

- No Histograma e KDE da Figura B.30 observam-se pelo menos 2 picos, indicativos de, pelo menos, $N = 2$ componentes do modelo GMM;
- Os gráficos da comparação de estimativas KDE e *Quantile-Quantile* (Figuras B.46 e B.62, respetivamente) demonstram que a distribuição do modelo GMM com $N = 2$ componentes é a distribuição mais simples que melhor se ajusta aos dados reais, principalmente nos quantis mais baixos;

- Na Tabela B.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer uma das distribuições relacionadas;
- As estatísticas dos testes AD e KS do modelo GMM com qualquer número de componentes correspondem $p - values$ elevados e superiores à distribuição Normal, o que indica que pode escolher-se o modelo GMM com $N = 2$ componentes, mais simples;
- Embora o indicador $l(\boldsymbol{\theta_{MLE}})$ seja o mais baixo, os indicadores BIC e AIC são os mais baixos de todas as distribuições do modelo GMM. Isto é indicativo de um bom ajustamento tendo em conta a menor complexidade da distribuição escolhida, face às restantes;

Após a escolha das distribuições que modelam os tempos de *picking* de cada turno do Cenário Otimista, a Tabela 3.5 apresentam os parâmetros associados a cada distribuição.

Tabela 3.5: Parâmetros das distribuições que modelam os tempos de *picking* dos turnos do cenário otimista. Os parâmetros não são arredondados para preservar o ajustamento feito.

Turnos do Cenário Otimista				
		Parâmetros		
Turno	**Distribuição**	**Média**	**Covariância**	**Peso**
ACR_LH_M2	GMM ($N = 2$)			
	Componente 1	109.90271452	325.21684403	0.6018544774451158
	Componente 2	153.31808406	589.44424788	0.3981455225548843
ACR_RH_M2	GMM ($N = 3$)			
	Componente 1	126.00318352	227.17531047	0.375640396005261
	Componente 2	164.44642145	364.05137691	0.22013435116190183
	Componente 3	100.48190269	239.52751191	0.4042252528328372
MM_M1	GMM ($N = 2$)			
	Componente 1	145.80317179	362.86104793	0.3615054463682051
	Componente 2	114.29974568	238.94000293	0.638494553631795
TF_M1	GMM ($N = 2$)			
	Componente 1	286.8748315	2157.39955639	0.4532357066060794
	Componente 2	376.98572317	2437.40400716	0.5467642933939206

3.2.2.4 Tempos de *Load*/*Unload* dos AMR

Os tempos de *Load*/*Unload* dos AMR foram recolhidos manualmente, utilizando um dos robôs construídos para o projeto. Seguindo novamente os passos estipulados, são identificados e eliminados os *outliers* através do respetivo gráfico *boxplot* (Figura C.1). Após eliminar os *outliers*, é calculada a dimensão, média e desvio padrão da amostra final (Tabela C.1). De seguida, é feita a estimativa não paramétrica da função densidade de probabilidade através da KDE (Figura C.2), passando também pelo ajustamento dos dados reais através do GMM e da biblioteca "Fitter" (Figura C.3). Os resultados dos testes e indicadores de ajustamento estão descritos na Tabela C.2, acompanhados pelos gráficos *Quantile-Quantile* (Figura C.4).

Na Tabela C.2, os $p - values$ dos testes AD e KS são superiores a α, independentemente da distribuição ajustada, o que indica existência de evidência estatística para não rejeitar a H_0 para qualquer distribuição ajustada. Através dos gráficos *Quantile-Quantile* (Figura C.4), observa-se que, de facto, todas as distribuições se ajustam aos dados reais na zona central da distribuição. Os indicadores de ajustamento revelam que o modelo GMM com $N = 2$ componentes é aquele cujo BIC e AIC são os menores, embora o $l(\boldsymbol{\theta_{MLE}})$ não seja o menor. Apesar do $p - value$ do teste KS não ser o mais elevado, as Figuras C.4a e C.3a indicam que esta distribuição se trata de um bom ajustamento aos dados reais, tanto na zona central como em ambas as caudas.

Após a escolha da distribuição que melhor se ajusta aos dados recolhidos dos tempos de *load/unload*, as Tabelas 3.6 apresenta os parâmetros associados a cada componente da distribuição escolhida.

Tabela 3.6: Parâmetros da distribuição que modela os tempos de *load/unload* dos AMR. Os parâmetros não são arredondados para preservar o ajustamento feito.

Parâmetros da distribuição dos tempos de *Load/Unload*				
		Parâmetros		
Dados	**Distribuição**	**Média**	**Covariância**	**Peso**
Load/Unload	GMM ($N = 2$)			
	Componente 1	13.79260335	0.0920707	0.3396457575441678
	Componente 2	14.30458779	0.05775911	0.6603542424558323

De forma a confirmar se as distribuições de *input* escolhidas para modelar os tempos do processo de *picking* e *load* e *unload* de *racks* pelos AMRs são significativamente melhores do que as distribuições escolhidas no estudo anterior, o passo seguinte seria a utilização do modelo de simulação já validado por Almeida (2022) para comparar o *output* do modelo quando utilizados cada um dos conjuntos de distribuições. Para o efeito, deverão ser seguidas as Etapas de Simulação descritas por Law (2013) (Figura 2.5).

Capítulo 4

Proposta de um Modelo de Calendarização Robusta

Neste capítulo propõe-se um novo modelo para a alocação de pedidos de movimentação de *racks* aos AMRs disponíveis. A abordagem atual, tal como descrita no Capítulo 3, utiliza o critério "*Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR) livre mais próximo da *rack*" para escolher o robô que irá satisfazer o pedido de movimentação. Esta abordagem puramente reativa por parte da frota de AMRs apresenta vantagens como a capacidade de manter o nível de serviço em linhas de produção/montagem cujos tempos de ciclo têm diferentes graus de variabilidade. No entanto, está também associada a desvantagens, como a possibilidade de falhas de comunicação entre AMRs ou a aceitação de um pedido por parte de um único AMR livre que se encontra mais longe do que um AMR mais próximo e que está prestes a terminar uma tarefa. A utilização de *triggers* para a emissão de pedidos de transporte é substituída por uma nova abordagem de alocação e aceitação de tarefas, que considera todos os AMRs, a sua localização, instante em que estão livres para iniciar uma nova tarefa, bem como fontes de incerteza tais como a ocorrência/duração de avarias dos robôs e duração das tarefas a desempenhar.

Propõe-se assim um modelo de Calendarização Robusta com recurso à Programação Linear Inteira Mista (MILP). Para o efeito, é necessário conceptualizar o problema de calendarização a resolver, bem como descrever as várias componentes do problema *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP) (função objetivo, variáveis de decisão, parâmetros ou coeficientes, e restrições).

4.1 Conceptualização do Problema

O projeto AGiLE pretende, a cima de tudo, responder ás necessidades de abastecimento da linha de montagem da Volkswagen Autoeuropa, Lda. (AE) com a satisfação de tarefas de movimentação de *racks*, vazias ou cheias, por parte de uma frota de AMRs. A metodologia de *Robust Scheduling* ou Calendarização Robusta (RS) proposta pressupõe a existência de um conjunto de n tarefas distintas, a executar por m robôs idênticos. A alocação de tarefas é feita

apenas aos m robôs disponíveis, ou seja, que tenham bateria suficiente e que não estejam avariados no momento do pedido de novo planeamento.

Este problema assemelha-se ao *Unrelated Parallel Machines Problem* ou Problema de Máquinas Paralelas não relacionadas (UPMP), que, tal como descrito por Pinedo (2007), pretende alocar um conjunto de n tarefas a um conjunto de m recursos idênticos, que as irão concluir sem interrupção das mesmas. Os recursos são considerados "não relacionados" pois o tempo de processamento de uma tarefa depende do recurso a que esta é alocada.

O problema tem como pressuposto a execução de uma tarefa de cada vez por cada recurso, critério satisfeito também pelos AMRs, dado que estes são capazes de movimentar apenas uma rack de cada vez, sem interromper a tarefa em execução até que esta seja cumprida. O objetivo passa por construir um planeamento robusto de n tarefas, onde cada tarefa é alocada a apenas um robô e onde são indicados os instantes de início e fim previstos da tarefa, bem como qual o robô que irá desempenhar determinada tarefa (exemplo na Figura 4.1). Portanto, trata-se de uma **Calendarização de Alocação Estática**, onde as tarefas serão alocadas a um recurso específico.

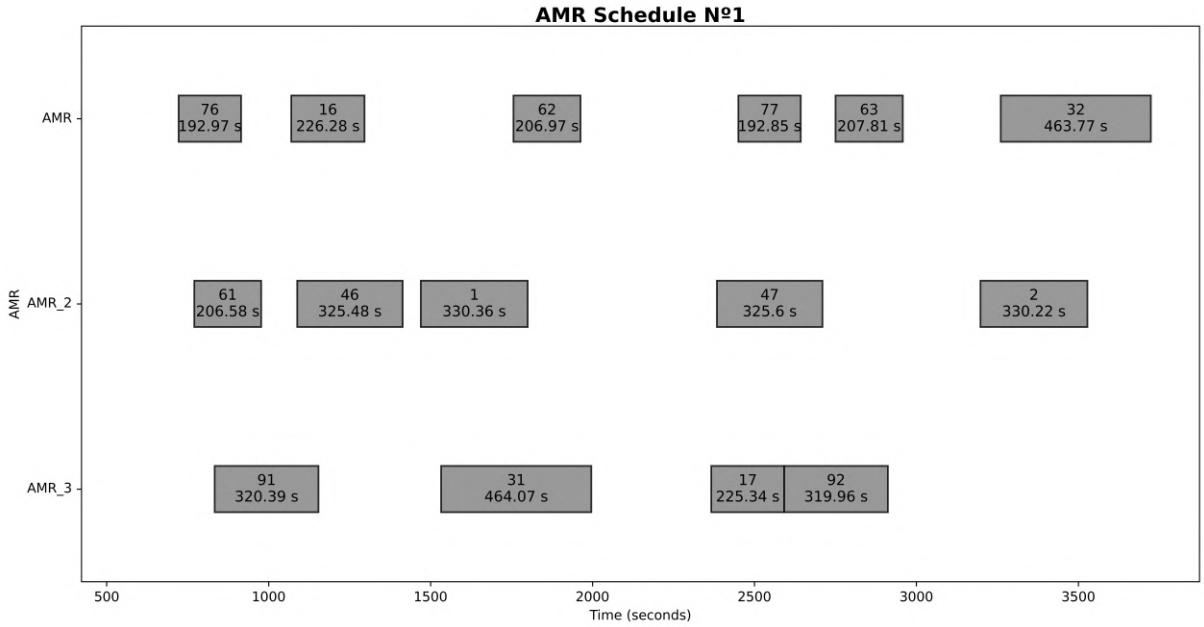


Figura 4.1: Exemplo de calendarização de $n = 15$ e $m = 3$ robôs. Os retângulos cinzentos correspondem a tarefas. O número inteiro representa o número de identificação da tarefa (*task_ID*), para efeitos de rastreabilidade aquando de uma futura implementação. O número real, em segundos, indica a duração da tarefa após a otimização.

Integrando uma abordagem Preditiva-Adaptativa, pretende-se que seja feito um primeiro **planeamento preditivo**, onde é usada a informação inicial para gerar as tarefas a executar. À medida que o sistema evolve e que é disponibilizada mais informação sobre o mesmo, é constantemente avaliada a possibilidade de ajustar o planeamento feito (**planeamento adaptativo**), realizando uma nova alocação de n tarefas (realocando tarefas já alocadas, mas ainda por executar, e novas tarefas). Se um planeamento é cumprido sem imprevistos e atrasos, ou seja, sem a necessidade de realocação de tarefas, é então gerado um novo planeamento para n tarefas e é feita uma nova iteração do processo descrito anteriormente (com um conjunto de n novas tarefas).

Segundo a notação $\alpha|\beta|\gamma$, introduzida por Graham et al. (1979), pode classificar-se este problema como ($R \mid resources \mid$"*Duração das Tarefas*"), tal que:

- $\alpha = R \rightarrow$ Problema de Máquinas Paralelas não relacionadas;
- $\beta = resources \rightarrow$ Existência de recursos limitados para a realização das n tarefas, sem *preemption* ou interrupção de uma tarefa, nem requisitos de precedência (não existe uma sequência de tarefas pré-definida);
- $\gamma =$ Duração das Tarefas (ver Equação 4.4);

4.1.1 Descrição das tarefas

A tarefa a desempenhar corresponde à movimentação de uma *rack* entre os Supermercado de manufatura (SUMA)s e *Point of Fit* (POF)s de um determinado componente (Figura 4.2). Nos SUMAs é realizado o processo de *picking*, diferente para cada componente e que, quando terminado, origina uma *rack* cheia que deverá ser movimentada até ao POF do respetivo componente. Já nos POF, as peças em cada *rack* são consumidas com uma cadência ou tempo de ciclo de 72 segundos por peça ($T_{cy} = 72\ s$). Esta é a cadência da linha de montagem determinada pelos responsáveis do projeto como a cadência máxima a que é consumida uma peça, permitindo ao AGiLE garantir o abastecimento da linha no cenário mais exigente. Quando uma *rack* fica vazia, esta deverá ser novamente movimentada até ao respetivo SUMA. Dado que cada *rack* é constituída por peças sequenciadas, é estritamente proibido retirar *racks* dos POF que não estejam vazias.

Independentemente do componente, podem definir-se dois tipos de tarefas durante o abastecimento (identificados na Figura 4.2), segundo a perspetiva de um robô:

- Movimentação de *racks* cheias;
- Movimentação de *racks* vazias.

Na Figura 4.2 é feito o paralelismo com os processos identificados na Figura 3.1, através da identificação dos SUMA's, POF's e processo de Pré-Montagem pela respetiva notação S, P e PM.

Para a realização de cada tipo de tarefa e execução do fluxo de processos identificado na Figura 4.2, o AMR alocado deverá deslocar-se ao local de recolha de uma *rack*, esperar que esta esteja pronta (se necessário), carregar-la (*load* da *rack*), deslocar-se ao local de descarga da *rack* e descarrega-la (*unload* da *rack*), assim que possível. Assim, na Figura 4.3 é descrito o fluxo de processos a ocorrer durante a alocação e execução de uma tarefa, da perspetiva de um robô. Na Tabela 4.1 são descriminadas as condições para a ocorrência das esperas identificadas e para a aceitação de tarefas, segundo o diagrama representado na Figura 4.3.

Os tipos de tarefas gerados pela combinação dos componentes passíveis de serem movimentados pelo AGiLE e dos tipos de movimentação de *racks* descritos anteriormente são apresentados na Tabela 4.2. O componente Módulos do Motor (MM), para além do respetivo SUMA e POF, conta ainda com o processo de Pré-Montagem como possível local de carga e descarga de *racks* cheias ou vazias.

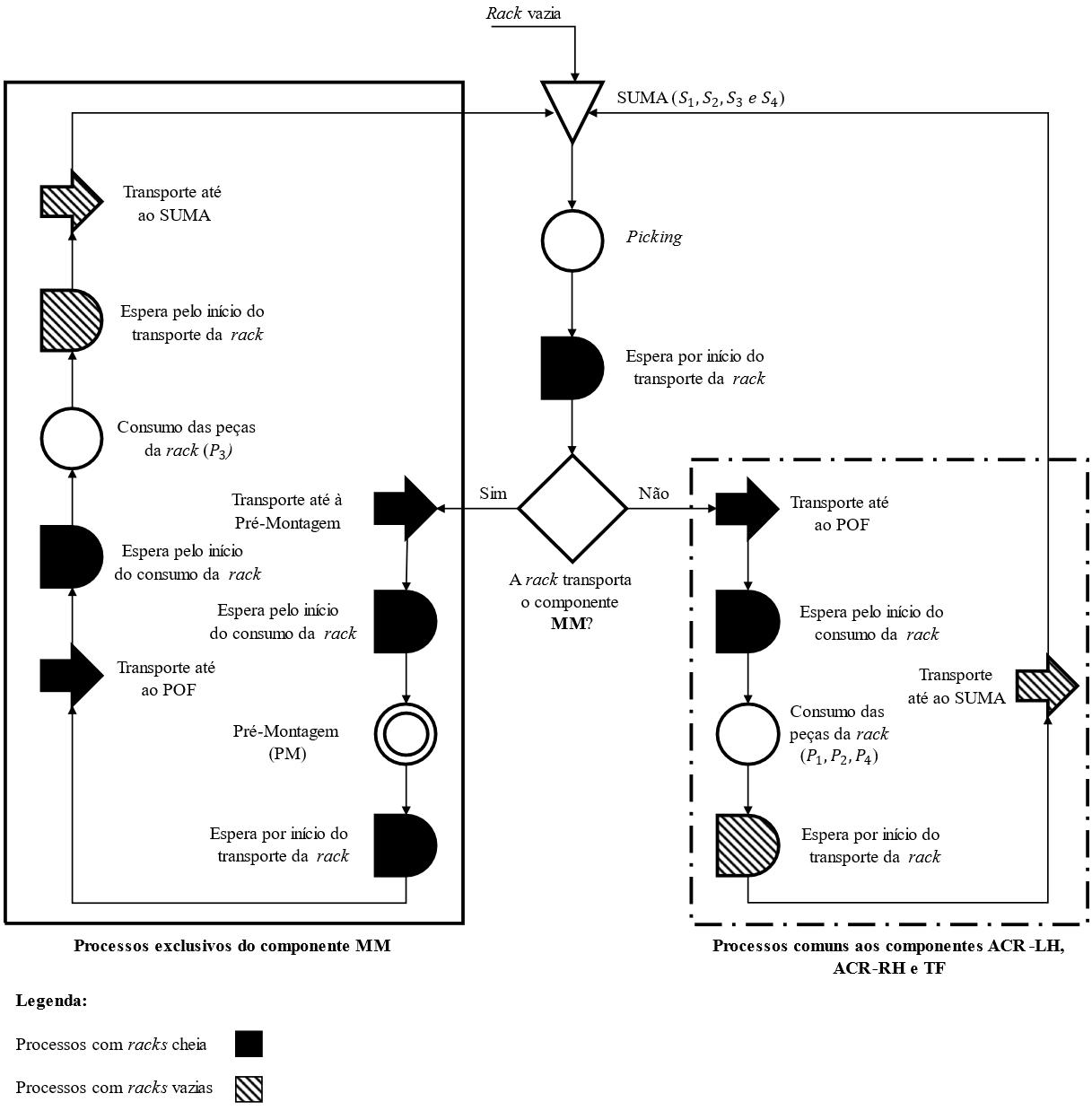


Figura 4.2: Diagrama de processo do abastecimento de cada componente.

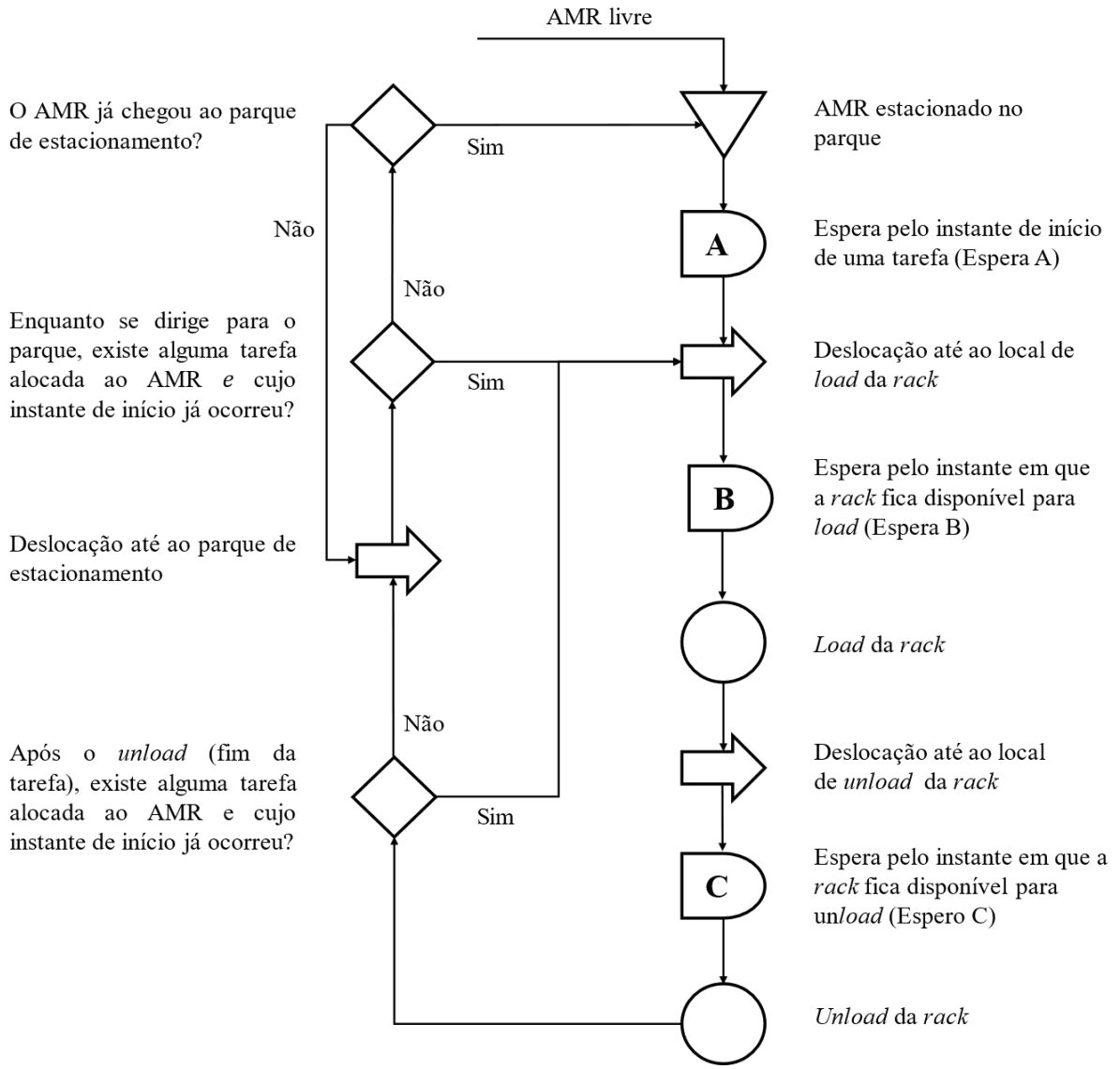


Figura 4.3: Diagrama de processo de uma tarefa de movimentação de *rack*. Na perspetiva do robô, a cada processo pode associar-se o instante ou intervalo de tempo que caracteriza ou limita esse evento, identificado como um parâmetro no problema MILP a construir.

Tabela 4.1: Condições para a ocorrência de cada espera e para a aceitação de uma tarefa.

Disjunção de Condições	**Espera A**	**Espera B**	**Espera C**	**Tarefa é aceite**
AMR está estacionado	X			
Picking não está terminado		X		
Rack ainda tem peças no POF		X		
Rack ainda não tem peças suficientes na Pré-Montagem		X		
Existência de espaço no local de descarga da *rack*			X	
AMR termina uma tarefa $\wedge$ Recebe uma nova tarefa				X
AMR está a caminho do parque de estacionamento $\wedge$ Recebe uma nova tarefa				X

Tabela 4.2: Tipos de tarefas a alocar. Estes dependem do componente a movimentar e do tipo ou estado de *rack* (vazia ou cheia), que, por sua vez, define o local de carga (*load*) e descarga (*unload*).

Tipo	**Componente**	**Tipo de *rack***	**Local de Carga**	**Local de Descarga**
1	ACR-LH	Cheia	SUMA	POF
2	TF	Cheia	SUMA	POF
3	ACR-LH	Vazia	POF	SUMA
4	TF	Vazia	POF	SUMA
5	MM	Cheia	SUMA	Pré-Montagem
6	MM	Cheia	Pré-Montagem	POF
7	MM	Vazia	POF	SUMA
8	ACR-RH	Vazia	POF	SUMA
9	ACR-RH	Cheia	SUMA	POF

4.1.2 *Deadlines* das tarefas

Todas as tarefas geradas terão as características associadas ao respetivo tipo de tarefa. Uma destas características é o seu *deadline*. É gerada uma tarefa quando se prevê que o POF tenha consumido todas as peças de uma *rack* e que é necessário a movimentação de uma *rack* cheia para satisfazer a necessidade de peças. Tendo em conta a cadência de 72 segundos, é possível prever de quanto em quanto tempo é necessária a movimentação de uma rack vazia e cheia.

No momento em que é pedida a alocação de n tarefas a m robôs (instante t_0), são retiradas n tarefas de um conjunto de tarefas previstas, por calendarizar. Esta lista é atualizada caso ocorram imprevistos que alterem os *deadlines* subsequentes, como atrasos no abastecimento face ao previsto. É atribuído um *deadline* (dd) previsto a cada tarefa, calculado consoante a capacidade dos POFs de cada componente no instante t_0 e do tempo de ciclo de 72 segundos por peça.

A Figura 4.4 demonstra o cálculo dos *deadlines* do componente Tampas e Filtros (TF), a título de exemplo. Nesta, se tt representar o tipo de uma tarefa (coluna "Tipo" da Tabela 4.2), considera-se $dd_{tt,a}$ o *deadline* da tarefa do tipo tt, para movimentar a *rack* a. No momento t_0, existem ainda x peças numa *rack* a, situada no POF, e onde cada *rack* tem capacidade para C peças. Assume-se que, neste caso, não existem *racks* de *buffer*, ou seja, deverá ser entregue uma *rack* cheia até ao instante em que uma *rack* no POF passa a estar vazia. Da mesma forma, essa mesma *rack* vazia deverá ser recolhida, no limite, no instante em que passou ao estado "vazia".

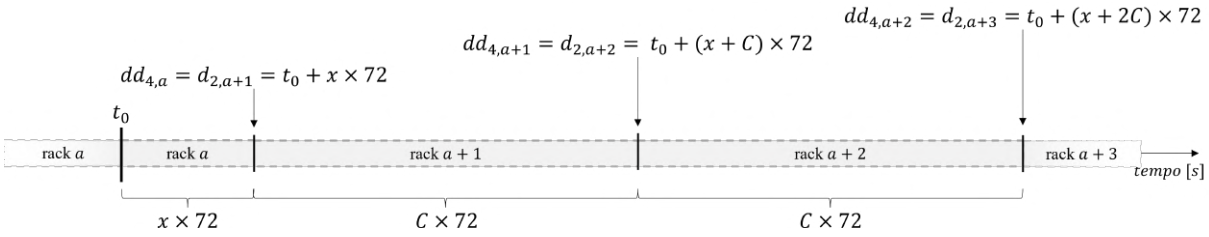


Figura 4.4: Cálculo dos instantes *deadline* previstos do componente TF, tendo em conta a capacidade C de uma *rack*, o número de peças por consumir POF (x) e o tempo de ciclo de 72 segundos por peça. Associados a este componente estão os tipos de tarefa 2 e 4. Enquanto a *rack* a é consumida, a *rack* cheia $a + 1$ (tarefa do tipo 2) deverá ser entregue até ao instante $dd_{2,a+1}$. Por sua vez, a *rack* a é totalmente consumida até ao mesmo instante e deverá ser recolhida (tarefa do tipo 4) no instante $dd_{4,a}$. A lógica é repetida para as *racks* subsequentes. A notação utilizada neste diagrama é apenas ilustrativa. Note-se que a notação para os *deadlines* de cada tarefa não inclui estes índices.

4.1.3 Fontes de Incerteza

Problemas de Calendarização Robusta consideram fontes de incerteza inerentes ao sistema para o qual se pretende obter um planeamento de operações resiliente. Estas estão intimamente ligadas não só aos agentes que interagem entre si e com o ambiente, mas também ao tipo de operações que surgem do normalmente funcionamento do sistema. Situando o problema em ambiente fabril, podem identificar-se vários elementos de origem externa, interna ou mesmo pontuais, que podem pôr em causa a movimentação de racks entre os SUMA e POF, impedindo o abastecimento das estação de trabalho e, consequentemente, provocar a paragem da linha de montagem por escassez de peças.

As fontes de natureza externa incluem todos os eventos imprevistos provenientes de entidades que não fazem parte da arquitetura do AGiLE (como fornecedores ou operadores da

linha de montagem). Por sua vez, as fontes de incerteza interna referem-se ao todos os eventos diretamente relacionados com entidades do AGiLE (robôs, *racks* e entidade supervisora). As fontes pontuais correspondem a eventos anormais, nomeadamente avarias tanto de elementos externos (como avarias nos SUMA/POF) como elementos internos (avarias de robôs), que necessitem de operações de manutenção corretiva.

A Figura 4.5 apresenta um Diagrama de *Ishikawa* com as fontes de incerteza recolhidas por observação do modelo de simulação já desenvolvido (Almeida, 2022), bem como por observação do sistema em teste, que influenciam direta e indiretamente o abastecimento de peças a cada estação de trabalho. Este diagrama exclui outras fontes de incerteza que possam surgir com o término da implementação e teste do AGiLE na AE.

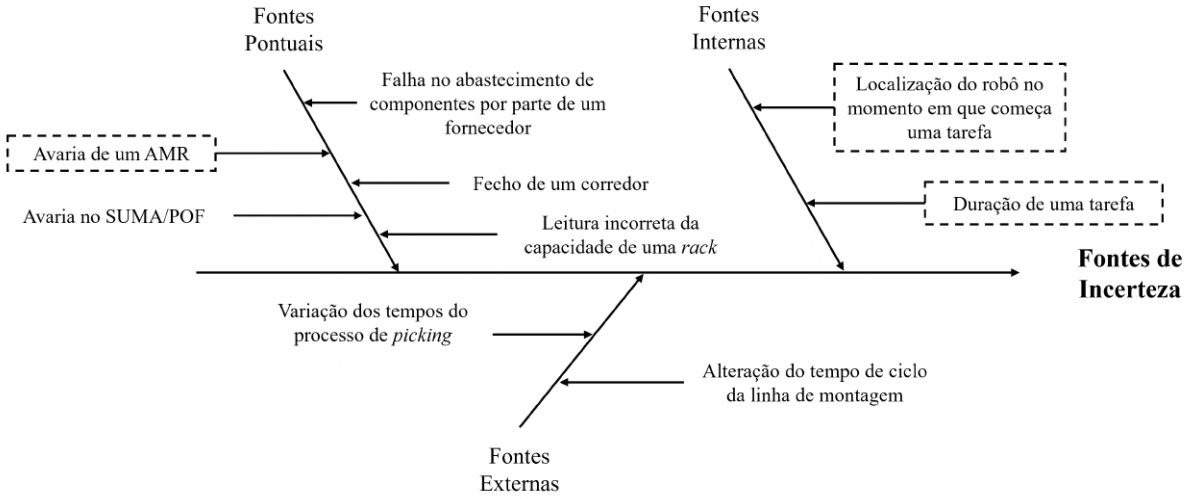


Figura 4.5: Fontes de incerteza associadas ao abastecimento por AMR, através do AGiLE. As fontes assinaladas são as escolhidas para integrar como fontes de incerteza no problema de calendarização, numa primeira abordagem, dada à facilidade de recolha de informação sobre estes eventos.

4.2 Formulação do Problema utilizando Programação Linear Inteira Mista

O problema de RS descrito anteriormente pode ser formulado como um problema de **Programação Linear**. Segundo Hillier e Lieberman (2015), tal como o nome indica, Programação Linear implica a utilização de um conjunto de expressões lineares para, neste caso, programar ou planear a alocação de n atividades (ou tarefas) a um conjunto limitado de m recursos da forma mais eficaz e eficiente possível. A forma como cada tarefa é alocada a cada recurso está limitada por um conjunto de **restrições lineares** (na forma de equações e inequações lineares) e é avaliada pela maximização ou minimização de uma ou mais **medidas de desempenho** escolhidas (denominadas Funções Objetivo). Todos estes elementos do problema são afetados por **parâmetros**, incluídos como coeficientes nas expressões lineares e que dependem exclusivamente da natureza do problema. A decisão final de qual tarefa é alocada a qual recurso, entre outras, é registada em **variáveis de decisão**. Em particular, este é um **problema de Programação Linear Inteira Mista** (MILP), dado que algumas das variáveis de decisão podem tomar apenas

valores inteiros positivos ou zero ($\mathbb{N}$), enquanto que outras podem tomar qualquer valor real positivo, incluindo o zero ($\mathbb{R}_0^+$).

A formulação do problema implica a garantia dos seguintes pressupostos (Hillier & Lieberman, 2015):

1. Proporcionalidade - A contribuição de cada variável de decisão na função objetivo e restrições lineares é proporcional ao valor da variável de decisão;
2. Aditividade - As funções lineares resultam da adição de termos com a contribuição de cada variável de decisão;
3. Divisibilidade - As variáveis de decisão podem tomar valores não inteiros, caso as restrições o permitam;
4. Certeza - Todos os parâmetros são constantes conhecidas.

Uma solução ótima do problema corresponde à definição do valor de cada variável de decisão que maximiza/minimiza a função ou funções objetivo e que pertença ao conjunto de soluções possíveis tendo em conta as restrições impostas pelo decisor.

A seguinte formalização matemática e respetiva notação do problema MILP são baseadas no modelo matemático proposto por Ghaleb e Taghipour (2023). São descritos os conjuntos, índices e parâmetros utilizados, bem como as variáveis de decisão, funções objetivo e restrições que limitam o conjunto de soluções possíveis.

4.2.1 Conjuntos, Índices, Parâmetros e Variáveis de Decisão

De acordo com o tipo de tarefas a alocar aos m robôs e com as características dos processos envolvidos no abastecimento, podem definir-se os conjuntos, índices, parâmetros e variáveis de decisão a utilizar na formalização da função objetivo e restrições do problema. Neste problema, os índices não estabelecem uma sequência de tarefas. São utilizados como identificação das tarefas para, posteriormente, estabelecer a sequência de tarefas que otimiza a Função Objetivo.

Conjuntos

M Conjunto de AMRs (Máquinas): $M = \{0, ..., m-1\}$
J Conjunto de tarefas: $J = \{0, ..., n-1\}$
L Conjunto de tarefas anterior a uma outra tarefa: $L = \{-1, ..., n-1\}$

Índices

i Índice dos AMRs: $i \in M$
j Índice das tarefas por alocar: $j \in J$
k Índice das tarefas a alocar após a tarefa j: $k \in J$
l Índice das tarefas a alocar antes de j: $l \in L$

Parâmetros

m	Número de AMRs disponíveis
n	Número de tarefas a alocar
t_{lj}^{c}	Tempo de movimentação do AMR até ao ponto de carga de uma *rack* da tarefa j, sabendo que l foi a última tarefa alocada ao AMR (em segundos)
t_{j}^{d}	Tempo de movimentação do AMR até ao ponto de descarga de uma *rack* da tarefa j (em segundos)
t_{ij}^{load}	Tempo de carga de uma *rack* do AMR i quando este está a desempenhar a tarefa j (em segundos)
t_{ij}^{unload}	Tempo de descarga de uma *rack* do AMR i quando este está a desempenhar a tarefa j (em segundos)
f_{ij}^{c}	Duração prevista de uma avaria do robô i durante a deslocação até ao ponto de carga da tarefa j (em segundos)
f_{ij}^{d}	Duração prevista de uma avaria do robô i durante a deslocação até ao ponto de descarga da tarefa j (em segundos)
p_{ilj}	Duração prevista, em segundos, da tarefa j quando alocada ao AMR i e se a última tarefa alocada ao mesmo robô for a tarefa l (Equação 4.1)
tt_j	Tipo da tarefa j (coluna "Tipo" na Tabela 4.2)
dl_j	Parâmetro binário onde, se $dl_j = 1$, a tarefa j precisa de *delay* após a recolha uma *rack* vazia. Caso contrário, se $dl_j = 0$, a tarefa não precisa de *delay*, nomeadamente após a recolha de uma rack cheia
dd_j	Prazo de entrega ou *deadline* de uma *rack* cheia ou recolha de uma *rack* vazia (em segundos)
tt_i^{last}	Tipo da última tarefa desempenhada pelo AMR i no último calendário (Tabela 4.2)
$time_i$	Instante em que a última tarefa alocada ao AMR i termina, no calendário anterior (em segundos)
Q	*Makespan* previsto tendo em conta o instante de término das últimas tarefas de cada AMR, no planeamento anterior, e o *deadline* mais distante do calendário atual (Equação 4.2).

A duração prevista de uma tarefa j é definida pela Equação 4.1:

$$p_{ilj} = t_{lj}^{c} + t_{ij}^{load} + t_{j}^{d} + t_{ij}^{unload} + f_{ij}^{c} + f_{ij}^{d}. \tag{4.1}$$

O parâmetro Q é definido pela Equação 4.2:

$$Q = 1000 \times \left\lceil (\max(dd_j) - \min(time_i))/1000 \right\rceil. \tag{4.2}$$

Os parâmetros f_{ij}^{c} e f_{ij}^{d} incorporam a fonte de incerteza pontual **"Avaria de uma AMR"**. Numa primeira abordagem, as distribuições probabilísticas que modelam o MTBF (*Mean Time Between Failure*) e MTTF (*Mean Time To Failure*) são substituídas por um modelo *Zero Inflated* (Zuur et al., 2009). Considerando uma determinada distribuição probabilística que modele a

duração de uma avaria de um robô (MTTR ou *Mean Time To Repair*), pode usar-se um modelo cuja probabilidade da duração de uma avaria ser nula é $1 - p$ e onde a probabilidade de ocorrer uma avaria com duração é determinada pela distribuição correta é p. A título de exemplo, se $p = 1\%$, existe 99% de probabilidade da duração de uma avaria do robô i ser nula. Significa também que existe 1% de probabilidade da duração de uma avaria do robô i ser diferente de zero e modelada pela distribuição que o decisor achar mais adequada.

Variáveis de Decisão

- x_{ij} Variável binária onde, se $x_{ij} = 1$, a tarefa j está alocada ao AMR i. Caso contrário $x_{ij} = 0$
- y_{ilj} Variável binária onde, se $y_{ilj} = 1$, a tarefa j é alocada ao AMR i, após a tarefa l. Caso contrário $y_{ilj} = 0$
- S_{ilj} Variável contínua que descreve o instante em que o AMR i inicia a tarefa j, se a última tarefa alocada ao robô for a tarefa l (em segundos)
- C_{ilj} Variável contínua que descreve o instante em que o AMR i chega ao **ponto de carga** de uma *rack* vazia ou ao **ponto de descarga** de uma *rack* cheia (em segundos)
- F_{ilj} Variável contínua que descreve o instante em que o AMR i termina a tarefa j, se a última tarefa alocada ao robô for a tarefa l (em segundos)

Tendo em conta o diagrama da Figura 4.3, as tarefas de movimentação de *racks* vazias e cheias podem ser representadas pelos diagramas das Figuras 4.6 e 4.7, respetivamente.

D_{ilj} não é utilizado no modelo. É usado apenas como abreviação da Equação 4.3 daqui em diante:

$$\mathrm{D_{ilj}} = \mathrm{dl_j} \times (\mathrm{t_{ij}^{load}} + \mathrm{t_j^d} + \mathrm{t_{ij}^{unload}} + \mathrm{f_{ij}^d}). \tag{4.3}$$

D_{ilj} é nulo sempre que uma *rack* precisa de *delay*, ou seja, numa tarefa de movimentação de *rack* vazia ($dl_j = 1$). Quando $dl_j = 1$, a duração $t_{ij}^{load} + t_j^d + t_{ij}^{unload} + f_{ij}^d$ é atrasada, para ocorrer apenas após o *deadline*, como é explicado nos próximos parágrafos.

Através do modelo de simulação já validado, observou-se que a Espera B (Figura 4.3) é aquela com maior impacto no abastecimento dos componentes, principalmente quando é movimentada uma *rack* vazia. É necessário que esta seja recolhida o quanto antes para que haja espaço junto à linha de montagem para uma nova *rack* cheia. Neste caso, quando um robô fazer *load* de uma *rack* vazia, é necessário que esta esteja completamente vazia, o que nem sempre acontece quando um robô chega a um dos POF's. As restantes Esperas A e C, dada o seu menor impacto no abastecimento, serão consideradas nulas, numa primeira abordagem.

Quando é transportada uma *rack* cheia, a Espera B não ocorre, pois o tempo de *picking* no SUMA e o tempo de ciclo da Pré-Montagem permitem que ambos os processos estejam terminados quando um robô chega para fazer *load* uma *rack* cheia. Assim, a Espera B não será considerada para a movimentação de *racks* cheias. Ainda neste tipo de tarefas, a Espera C poderá ocorrer se não existirem condições para fazer o *unload* da *rack* no ponto de descarga. Neste caso, a Espera C é considerada no final deste tipo de tarefas.

Para uma *rack* vazia, o *deadline* da tarefa corresponde a momento em que a *rack* fica disponível no ponto de carga. Portanto, a duração da deslocação desde o momento em que é aceite a tarefa (S_{ilj}) até ao local de carga é, no mínimo, $t^c_{lj} + f^c_{ij}$, ou seja, $p_{ilj} - D_{ilj}$ segundos. Pode dizer-se que este tipo de tarefas precisam de um *delay* ($dl_j = 1$). No instante em que o robô chega ao ponto de carga (C_{ilj}), este pode ou não ficar à espera que a *rack* fique disponível (Espera B). Assim, o intervalo entre C_{ilj} e dd_j não está limitado por nenhum parâmetro e o robô apenas consegue fazer *load* no instante dd_j. Quando o robô faz o *load* da *rack*, desloca-se até ao ponto de descarga. Esta deslocação demora, no mínimo, D_{ilj} segundos, terminando no instante $F_{ilj} = dd_j + D_{ilj}$.

Para uma *rack* cheia, a estrutura da tarefa tem em conta a Espera C, ou seja, a impossibilidade de entregar a *rack* assim que o AMR chega ao local de descarga. Considera-se que o robô consegue realizar a totalidade da duração prevista (p_{ilj}) sem interrupção, o que se traduz num tipo de tarefa que não precisa de *delay* ($dl_j = 0$). Quando chega ao local de descarga da *rack* cheia, no instante C_{ilj}, consoante a disponibilidade do local, o AMR pode fazer *unload* da *rack* o mais próximo do deadline possível.

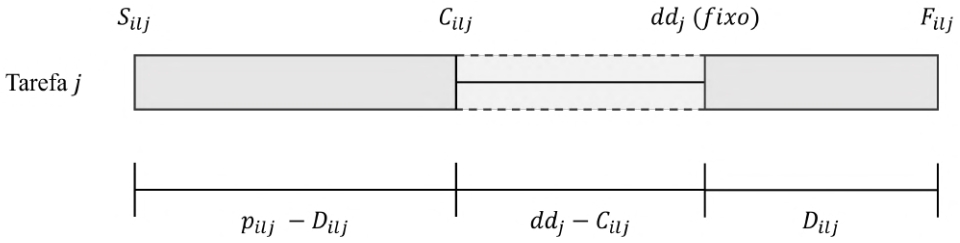


Figura 4.6: Representação de uma tarefa de movimentação de *rack* vazia.

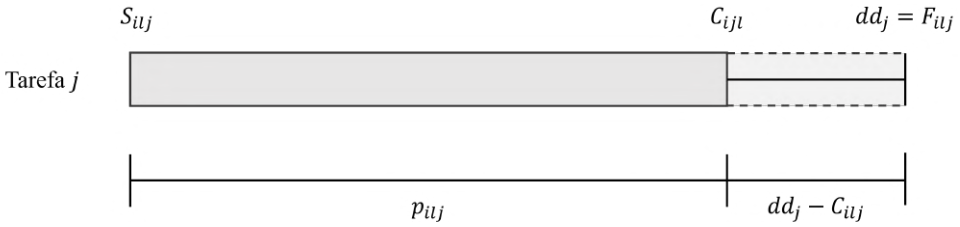


Figura 4.7: Representação de uma tarefa de movimentação de *rack* cheia.

Existem parâmetros e variáveis de decisão onde a tarefa j depende da tarefa l. Isto ocorre pois a localização onde o robô aceita a tarefa j depende do local onde o robô terminou a última tarefa l. Dependendo do tipo de tarefa e componente a movimentar, a tarefa anterior l pode terminar em locais diferentes da linha de montagem, pelo que a deslocação até ao ponto de carga da tarefa atual j será diferente. A partir do momento que um robô faz o *load* no ponto de carga, o ponto de descarga é conhecido, tal como a duração média da deslocação até ao mesmo, portanto já não depende da tarefa anterior. Esta é a razão pela qual, por exemplo, o parâmetro t^c_{lj} depende da tarefa anterior, mas o parâmetro t^d_j já não. Este tipo de parâmetros e variáveis de decisão incorporam as fontes de incerteza **"Localização do robô"** e **"Duração de uma tarefa"**, de forma a alocar um robô cuja localização seja mais benéfica para a satisfação de uma tarefa com a menor duração possível, sem atrasos ou tempos de espera junto aos pontos de carga e descarga de uma *rack*.

Esta dependência entre tarefas faz com que, entre dois planeamentos seguidos, a primeira

tarefa de cada robô dependa do tipo da última tarefa realizada pelo mesmo robô (tt_i^{last}), no planeamento anterior. Nos restantes parâmetros, quando se quer relacionar a primeira tarefa do calendário atual e última tarefa do último calendário executado, é utilizado o índice $l = -1$ para representar, no planeamento atual, uma tarefa que não pertence ao conjunto de tarefas atual J, mas que influência a duração da primeira tarefa de cada robô.

A Figura 4.8 apresenta um exemplo de solução ótima para a alocação de $n = 3$ tarefas a $m = 2$ robôs e onde existe dependência da duração p_{ilj} da tarefa anterior alocada ao robô. Repare-se que cada robô i executa uma tarefa j de cada vez e cada tarefa j é executada por um único robô i, como pretendido.

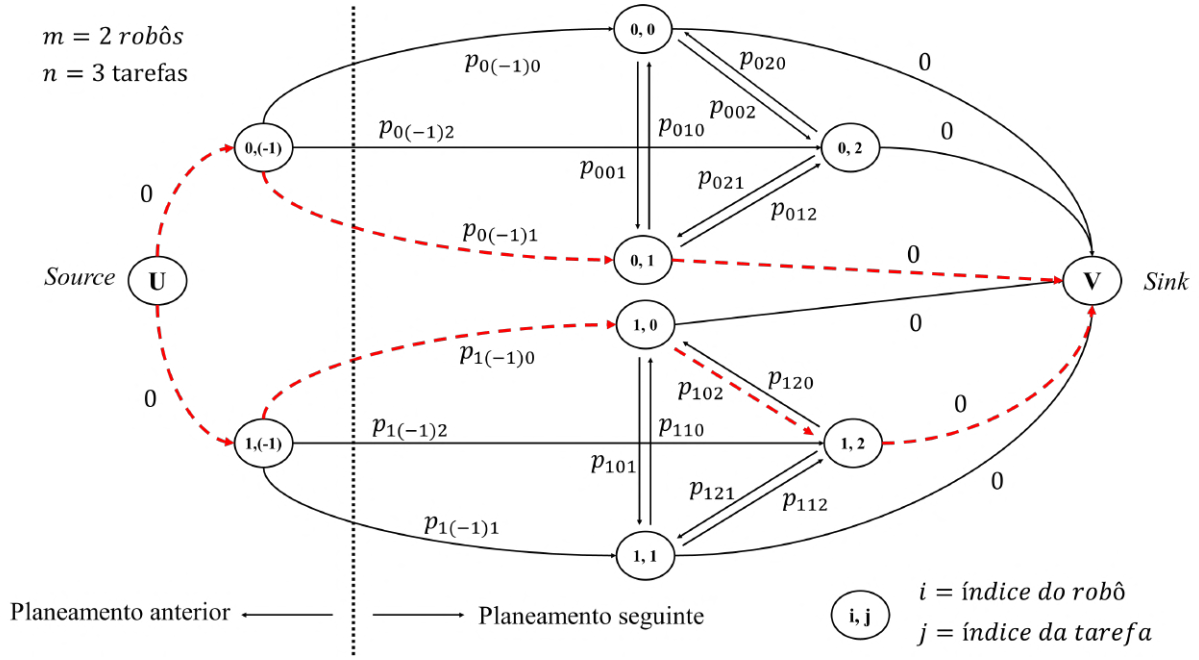


Figura 4.8: Representação em grafo da alocação de $n = 3$ tarefas a $m = 2$ robôs. A tracejado largo vermelho está destacada a sequência de tarefas ótima que minimiza a função objetivo. Adaptado de (Pinedo, 2016, p.185)

Dado que esta é uma representação de um planeamento que ocorreu na sequência de outros planeamentos anteriores, todas as arestas referentes a estes planeamentos não são representadas e o presente grafo é iniciado num nó *Source*. Este tipo de nós têm apenas arestas que saem em direção a outros nós, por isso podemos dizer que tem grau de entrada 0. Por sua vez, prevê-se que este exemplo seja seguido de outro planeamento do qual não irá depender, por isso todos os nós das possíveis últimas tarefas a executar por cada robô i deverão entrar num nó *Sink*, cujo grau de saída é 0 (Wilson, 2009). Na realidade, as últimas tarefas de cada robô estarão sempre ligadas às primeiras tarefas dos robôs no planeamento seguinte. No entanto, os nós *Sink* e *Source* são uma abstração desses planeamentos anteriores e posteriores, cujas restantes tarefas não influenciam o planeamento atual (à exceção das últimas tarefas de cada robô no planeamento anterior, representadas na Figura 4.8 pelos nós $(0, -1)$ e $(1, -1)$).

A duração de uma tarefa p_{ilj} corresponde ao peso da aresta entre dois nós (i, l) e (i, j) (com essa direção). Cada robô tem associado uma última tarefa com o índice $j = -1$, pertencente ao planeamento anterior, que influencia a duração da tarefa seguinte, ou seja, a primeira tarefa de cada robô no planeamento seguinte. No planeamento anterior, $j \neq -1$ e apenas existe o valor

$l = -1$ no novo calendário. Como todas as $n = 3$ tarefas terão de ser alocadas a um robô i, deverão fazer parte do caminho a tracejado 3 nós com diferentes índices j. Não podem existir dois nós com o mesmo índice j pois uma tarefa não pode ser alocada a dois robôs diferentes. Como existem mais tarefas do que robôs, duas tarefas serão alocadas sequencialmente ao mesmo robô. É o caso do robô $i = 1$, que irá executar as tarefas $j = 0$ e $j = 2$ neste exemplo.

A execução desta alocação de $n = 3$ tarefas a $m = 2$ AMRs é exemplificada na Figura 4.9, utilizando a notação utilizada na Figura 3.1, no Capítulo 3.

4.2.2 Função Objetivo

Dada a estrutura de cada tipo de tarefa apresentado e sabendo que se pretende um planeamento que aloque n tarefas a m robôs da forma mais eficaz e eficiente possível, pretende-se escolher função objetivo que:

1. Diminuam o tempo de espera de um robô junto aos pontos de carga e descarga de uma *rack*;
2. Evitem a entrega de racks após os *deadlines* previstos;
3. Minimizem o tempo de execução de uma tarefa, tendo em conta o AMR à qual esta está alocada e a tarefa que lhe foi alocada antes.

Desta forma, sugere-se a utilização da função objetivo "Duração das Tarefas"(Equação 4.4).

$$\min \quad \sum_{i=0}^{m-1} \sum_{l=-1}^{n-1} \sum_{j=0}^{n-1} F_{ilj} - S_{ilj}, \qquad (l \neq j) \tag{4.4}$$

Esta função permite não só a minimização da duração de cada tarefa como também, por consequência, a minimização do tempo de espera (*waiting time*) caracterizado por $dd_j - C_{ilj}$. Maiores tempos de espera resultam em AMRs alocados e com tempos de ócio que não contribuem para a satisfação da tarefa, ou seja, a entrega de um *rack* no local adequado. Estes desperdícios contribuem também para um maior consumo energético por tarefa, aumentando o número de vezes que o robô se dirige ao parque de estacionamento para carregar a bateria, ficando indisponível para executar tarefas até que tenha bateria suficiente.

A existência de *deadlines* fixos implica que o *makespan* ou o tempo de conclusão (*completion time*), por exemplo, não sejam as funções objetivo mais adequadas, embora sejam funções bastante comuns neste tipo de problemas.

4.2.3 Restrições

O conjunto de soluções possíveis para o presente problema MILP está condicionado por um conjunto de restrições. Estas restrições dependem exclusivamente da natureza do problema, das suas características e pressupostos. Podem definir-se vários tipos de restrições lineares (Liu & MacCarthy, 1997):

- Restrições de Capacidade - Controlam a alocação das tarefas aos recursos disponíveis;

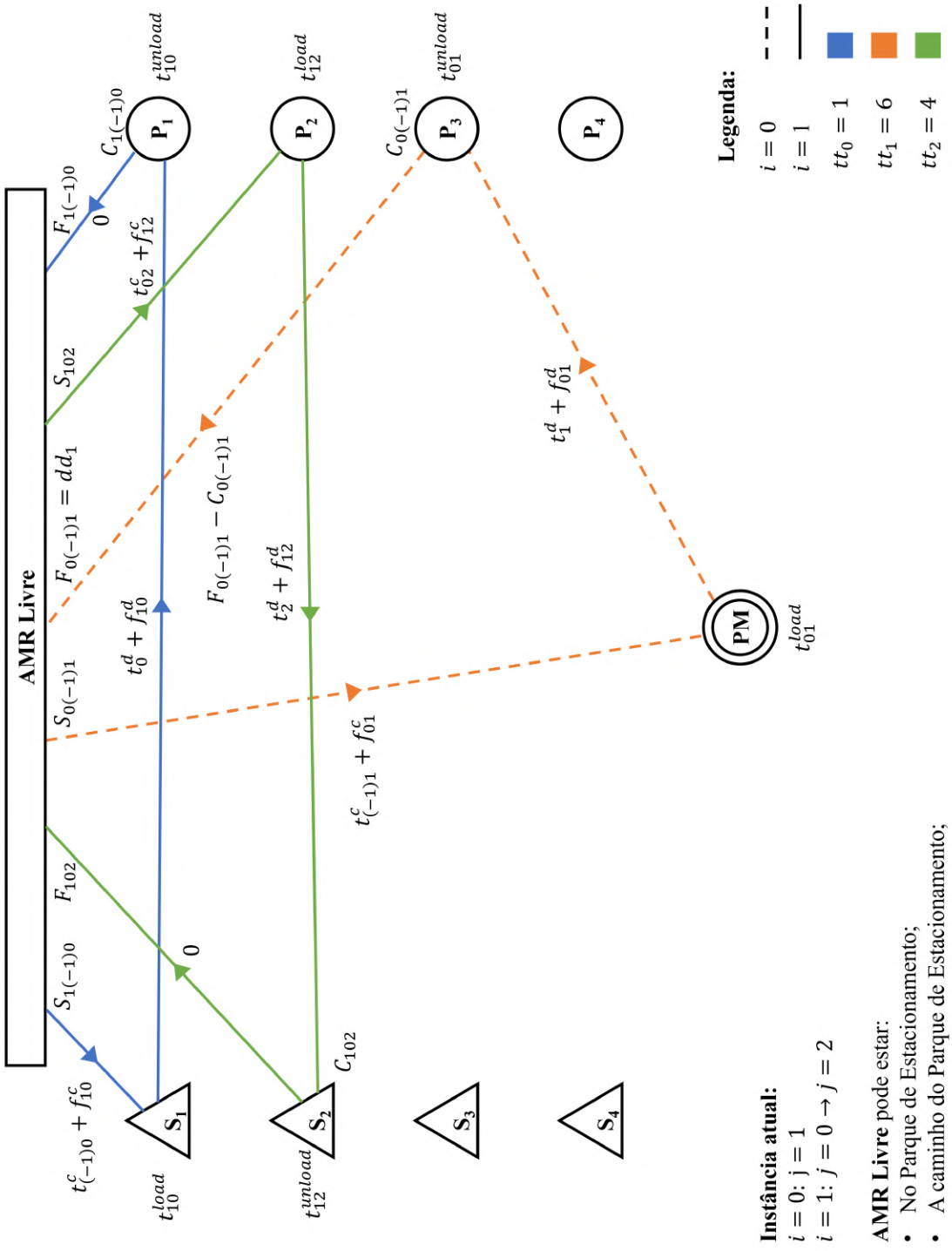


Figura 4.9: Representação em grafo da execução do planeamento gerado pela alocação de $n = 3$ tarefas a $m = 2$ robôs.

- Restrições de Precedência - Definem a duração prevista de uma tarefa, tendo em conta a precedência de uma tarefa alocada antes;
- Restrições de Sequenciamento - Garantem que não existe sobreposição de tarefas alocadas ao mesmo recurso;
- Restrições de Integralidade - Definem a natureza discreta ou continua de cada variável de decisão.

Cada restrição linear é construída com base em pressupostos do abastecimento, robôs e/ou do planeamento. As condições $(l \neq j)$, $(j \neq k)$ e $(k \neq l)$ garantem que apenas existe relação entre tarefas diferentes. Uma tarefa não pode ser alocada mais do que uma vez, portanto nenhum dos índices referentes a tarefas podem ser iguais.

Restrições de Capacidade

Uma tarefa corresponde a uma única movimentação de uma *rack* e que não é repetida, portanto uma tarefa j é alocada a um só AMR i (Equação 4.5):

$$\sum_{i=0}^{m-1} x_{ij} = 1, \qquad \forall\, j = \{0, ..., n-1\} \tag{4.5}$$

Quando o número de tarefas n é superior ao número de robôs m $(n \geq m)$, um AMR i tem, pelo menos, uma tarefa j alocada (Equação 4.6):

$$\sum_{j=0}^{n-1} x_{ij} \geq 1, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ (n \geq m) \end{array} \tag{4.6}$$

Um AMR i tem apenas uma primeira tarefa j em cada planeamento gerado e que irá depender da última tarefa realizada pelo robô i $(l = -1)$, na planeamento anterior. No entanto, a restrição a usar depende se o número de tarefas n é superior ao número de robôs m ou não. Se $n \geq m$, utiliza-se a Equação 4.7:

$$\sum_{j=0}^{n-1} y_{i(-1)j} = 1, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ (n \geq m) \end{array} \tag{4.7}$$

Caso contrário, é utilizada a Equação 4.8:

$$\sum_{j=0}^{n-1} y_{i(-1)j} \leq 1, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ (n < m) \end{array} \tag{4.8}$$

Por consequência, uma tarefa j só pode ser a primeira tarefa de um dos AMR i, quando necessário. Se $n > m$ é utilizada a Equação 4.9:

$$\sum_{i=0}^{m-1} y_{i(-1)j} \leq 1, \qquad \begin{array}{c} \forall\, j = \{0, ..., n-1\} \\ (n > m) \end{array} \tag{4.9}$$

Caso contrário, é utilizada a Equação 4.10:

$$\sum_{i=0}^{m-1} y_{i(-1)j} = 1, \qquad \begin{array}{c} \forall\, j = \{0, ..., n-1\} \\ (n \leq m) \end{array} \tag{4.10}$$

Restrições de Precedência

A duração de uma tarefa j (p_{ilj}) é calculada pela Equação 4.11. Se a tarefa j estiver alocada ao robô i após a tarefa l ($y_{ilj} = 1$), o valor da variável p_{ilj} é diferente de zero e igual à duração prevista para essa tarefa. Caso contrário, se $y_{ilj} = 0$, a variável p_{ilj} toma o valor zero.

$$p_{ilj} = y_{ilj} \times (t_{lj}^{c} + t_{ij}^{load} + t_{j}^{d} + t_{ij}^{unload} + f_{ij}^{c} + f_{ij}^{d}), \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.11}$$

O instante S_{ilj} deve ser superior ao instante que a última tarefa alocada ao AMR i foi terminada ($time_i$), no planeamento anterior (Equação 4.12). Se a tarefa j estiver alocada ao robô i após a tarefa l ($y_{ilj} = 1$), o valor da variável S_{ilj} é diferente de zero e superior a $time_i$. Caso contrário, se $y_{ilj} = 0$, a variável S_{ilj} é nula.

$$S_{ilj} \geq y_{ilj} \times time_i, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.12}$$

O intervalo entre o início da tarefa (S_{ij}) e a chegada ao ponto de carga de uma *rack* vazia ou descarga de uma *rack* cheia (C_{ij}) não excede a duração do tempo de processamento da tarefa p_{ilj} (Equação 4.13). Se a tarefa j estiver alocada ao robô i após a tarefa l ($y_{ilj} = 1$), a diferença $C_{ilj} - S_{ilj}$ é diferente de zero e igual à duração prevista. Caso contrário, se $y_{ilj} = 0$, a diferença $C_{ilj} - S_{ilj}$ toma o valor zero. Dependendo do tipo de tarefa, a diferença $C_{ilj} - S_{ilj}$ pode ser menor ou maior consoante a necessidade de *delay* ($dl_j = 1$) ou não ($dl_j = 0$).

$$C_{ilj} - S_{ilj} = p_{ilj} - y_{ilj} \times dl_j \times (t_{ij}^{load} + t_{j}^{d} + t_{ij}^{unload} + f_{ij}^{d}), \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.13}$$

O instante de término da tarefa j (F_{ilj}) é igual ou superior ao instante de início da mesma tarefa S_{ilj}, quando j é alocada ao robô i e após a tarefa l (Equação 4.14):

$$F_{ilj} - S_{ilj} \geq 0, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.14}$$

O instante C_{ilj} deverá ser menor ou igual ao *deadline* da tarefa (dd_j), quando j é alocada ao robô i e após a tarefa l (Equação 4.15).

$$C_{ilj} - dd_j \leq 0, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.15}$$

O instante de término de uma tarefa j (F_{ilj}) é igual ou superior ao *deadline* dd_j (Equação 4.16). Se a tarefa j estiver alocada ao robô i após a tarefa l ($y_{ilj} = 1$), a variável F_{ilj} é diferente de zero e igual ao instante previsto. Caso contrário, se $y_{ilj} = 0$, a variável F_{ilj} toma o valor zero. Dependendo do tipo de tarefa, a variável F_{ilj} pode ser igual a dd_j (se $dl_j = 0$) ou superior (se $dl_j = 1$).

$$F_{ilj} = y_{ilj} \times (dd_j + dl_j \times (t_{ij}^{load} + t_j^d + t_{ij}^{unload})), \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.16}$$

As Equações 4.11, 4.12, 4.13 e 4.16 utilizam a variável y_{ilj} para anular as variáveis p_{ilj}, S_{ilj}, C_{ilj} e F_{ilj} sempre que a tarefa j não é alocada ao robô i, após a tarefa l. Isto significa que o valor da função objetivo é apenas afetado pelas tarefas alocadas. Isto evita que se multiplique a variável y_{ilj} pelas variáveis na função objetivo.

Restrições de Sequenciamento

De forma a sequenciar um sub-conjunto de tarefas num AMR i, é necessário garantir que não existem sobreposições entre tarefas, dado um AMR não pode realizar duas tarefas ao mesmo tempo. Mais ainda, como a duração de uma tarefa k depende do tipo da tarefa anterior j, quando ambas estão alocadas ao mesmo robô i, são necessárias as variáveis binárias x_{ij}, x_{ik}, y_{ilj} e y_{ijk} para garantir que as tarefas j (dependente de uma tarefa l) e k (dependente da tarefa j) estão alocadas ao mesmo robô e que a ordem de execução das tarefas é $l \rightarrow j \rightarrow k$.

Os instantes S_{ijk} e F_{ijk} não dependem um do outro. Por exemplo, se uma tarefa tiver duração p_{ijk} e um instante de fim F_{ijk}, o instante de início não é calculado através da diferença $F_{ijk} - p_{ijk}$, mas sim através do instante C_{ijk}. Por isso, é necessário um par de Restrições de Sequenciamento para F_{ijk} e outro para S_{ijk}.

$$S_{ijk} - S_{ilj} + Q \times (4 - x_{ij} - x_{ik} - y_{ilj} - y_{ijk}) \geq p_{ilj}, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ \forall\, k = \{0, ..., n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{4.17}$$

$$S_{ijk} - S_{ilj} + Q \times (3 - x_{ij} - x_{ik} - y_{ilj} + y_{ijk}) \geq p_{ijk}, \qquad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ \forall\, k = \{0, ..., n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{4.18}$$

$$F_{ijk} - F_{ilj} + Q \times (4 - x_{ij} - x_{ik} - y_{ilj} - y_{ijk}) \geq p_{ijk}, \quad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ \forall\, k = \{0, ..., n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{4.19}$$

$$F_{ilj} - F_{ijk} + Q \times (3 - x_{ij} - x_{ik} - y_{ilj} + y_{ijk}) \geq p_{ilj}, \quad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ \forall\, k = \{0, ..., n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{4.20}$$

As Restrições de Sequenciamento utilizam o método *Big M* (Hillier & Lieberman, 2015, p. 115). Neste caso, a constante Q é suficientemente grande para controlar o valor de uma variável ou de uma operação entre variáveis quando necessário, ou seja, quando as variáveis binárias indicam a ocorrência (ou não ocorrência) de determinada condição.

Restrições de Integralidade

Devido à natureza temporal das variáveis contínuas $p_{ilj}, S_{ilj}, C_{ilj}$ e F_{ilj}, estas podem tomar qualquer valor real positivo ou zero ($\mathbb{R}_0^+$) (Equação 4.21).

$$p_{ilj}, S_{ilj}, C_{ilj}, F_{ilj} \geq 0, \quad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.21}$$

As variáveis x_{ij} e y_{ilj} são variáveis binárias (definidas pela Equação 4.22), que informam sobre a ocorrência de determinado evento ou característica.

$$x_{ij}, y_{ilj} \in \{0, 1\}, \quad \begin{array}{c} \forall\, i = \{0, ..., m-1\} \\ \forall\, l = \{-1, ..., n-1\} \\ \forall\, j = \{0, ..., n-1\} \\ (l \neq j) \end{array} \tag{4.22}$$

4.2.4 Implementação do *solver* Gurobi (GRB)

Para resolver o problema MILP apresentado é utilizado o Gurobi Optimizer (GRB), um software do tipo *solver*, que permite resolver problemas (lineares ou quadráticos) e obter um conjunto de soluções viáveis e uma solução ótima, sempre que possível. A empresa *Gurobi Optimization* foi fundada em 2008 com a missão de tornar a otimização uma ferramenta para a tomada de decisão mais "rápida e inteligente", em mais de 40 indústrias (Gurobi Optimization, 2023b).

Este *solver* está acessível através das linguagens de programação C, C++, Java, Microsoft.NET, *Python*, MATLAB e R (Gurobi Optimization, 2023a). Outros potenciais *solvers*, como o *IBM ILOG*

CPLEX Optimizer, poderão ser testados juntamente com o GRB, dada a sua vasta utilização neste tipo de problemas. A rápida obtenção da licença académica e a implementação do modelo formulado através da sua interface intuitiva permitiu escolher o GRB como o *solver* a utilizar.

Após o desenvolvimento do modelo, é necessário que seja testado e validado, através do modelo de *Discreet Event Simulation* ou Simulação de Eventos Discretos (DES) construído e validado por Almeida (2022), utilizando a ferramenta *Flexsim*. Esta permite a integração de funções e algoritmos através de *Python*, C ou C++. De entre a lista de linguagens através das quais se pode aceder ao GRB, *Python* é aquela que oferece maior versatilidade e aprendizagem intuitiva para implementar o modelo proposto no simulador *Flexsim*, bem como num software próprio para a calendarização de robôs, assim que testado. Um *survey* de 2022, resultado da colaboração entre a Python Software Foundation e JetBrains (2022), revela que 85% dos inquiridos utilizam *Python* como a sua linguagem de programação primária, tanto em projetos profissionais como pessoais, nas áreas de *Data Science*, *Web Development* e *Machine Learning*.

A programação e teste do *script* em *Python* (versão 3.10.4) é feita no ambiente de desenvolvimento integrado (*Integrated Development Environment* ou IDE) *PyCharm* (versão 2022.3.3, Professional Edition), instalada num computador HP Laptop 15s-fq2xxxn com o sistema operativo Microsoft Windows 11 Home, processador *Intel Core i7-1165G7* (11th Gen, 2.80GHz, 2701 Mhz, 4 núcleos e 8 processadores lógicos) e 12GB de RAM. O *solver* GRB instalado é a versão 10.0.2.

O algoritmo utilizado para a otimização dos modelos através do GRB depende do tipo de modelo (Gurobi Optimization, 2023a):

- Modelos contínuos → Algoritmo *Simplex* ou Algoritmo *Barrier*;
- Programação Inteira Mista (Linear ou Quadrática) → Algoritmo *Branch-and-Cut*.

O problema apresentado é um problema MILP, pelo que é usado o Algoritmo *Branch-and-Cut* para encontrar um conjunto de soluções sub-ótimas e uma solução ótima comprovada. Este algoritmo é a combinação dos algoritmos *Cutting Planes* e *Branch-and-Bound* e é muito utilizado na a otimização de problemas combinatoriais, como é o caso do Problema do Caixeiro Viajante (*Traveling Salesman Problem* ou TSP). Utiliza uma estrutura de dados chamada *Search Tree* ou Árvore de Procura, composta por nós e ramificações (do problema) geradas pelos diferentes valores que uma variável pode ter (Padberg & Rinaldi, 1991, p. 82).

O GRB permite o ajuste do *gap* máximo (Figura D.2, linha 42), em percentagem, entre o valor da função objetivo da melhor solução encontrada (melhor Bound) e o valor da função objetivo ótima. Neste caso, o parâmetro *gap* do *solver* será 0, para que apresente apenas soluções ótimas (Gurobi Optimization, 2023a, p. 843).

Quando o *solver* inicia a otimização, primeiro é feito um *presolve* do modelo (Figuras D.7 e D.8, linhas 399-437). Este obtém um modelo equivalente, mas onde foi reduzida a complexidade do modelo original, garantindo que a viabilidade/inviabilidade do modelo permanecem intactas e que, caso exista uma solução ótima, o valor ótimo da função objetivo permaneça idêntico. O *Presolve* tem duas fases distintas (Achterberg et al., 2020):

1. *Root Presolve* - Rotina de *presolve* aplicada ao problema associado ao nó raiz da árvore de procura do algoritmo *Branch-and-Bound*. É aquele com maior impacto na performance do

solver na otimização do problema, pois aplica reduções ao nível de cada restrição, variável, conjunto de restrições, conjunto de variáveis e reduções ao nível do problema na sua totalidade.

2. *Node Presolve* - Rotina de *presolve* aplicada aos sub-problemas ao longo da árvore de procura do algoritmo *Branch-and-Bound*. Menos agressivo que o *Root Presolve*, pois apenas estreita os *bounds* (limites inferior e superior) das variáveis do sub-problema em cada nó.

De forma a analisar o sub-problema após o *presolve*, é possível consultar a representação linear desse problema através dos ficheiros *.lp* e *.mps*, sendo o ficheiro *.lp* mais intuitivo (Figura D.8, linhas 436 e 437).

O *solver* faz um *log* do progresso do algoritmo *Branch-and-Cut* a cada 5 segundos para informar o utilizador sobre o estado da otimização (valor *default*, que pode ser alterado através do parâmetro *DisplayInterval*) (Gurobi Optimization, 2023a, p. 936). Este informa sobre o valor da função objetivo da melhor solução até ao momento e da melhoria da solução ao longo da procura em árvore, com recurso ao *gap* relativo entre o valor da função objetivo ótimo e o valor da função objetivo da melhor solução até ao momento (melhor *Bound*).

Após a otimização do modelo (Figura D.8, na linha 441), este poder ser viável ou *feasible*, inviável ou *infeasible*, ilimitado ou *unbounded*, entre outros estados (Gurobi Optimization, 2023a, p. 890). Se o modelo for viável, é possível consultar o valor final de cada variável, bem como ambos os lados de cada restrição (Figura D.9, nas linhas 491 e 492). Caso seja inviável, é possível consultar quais as restrições que são infringidas (direta ou indiretamente, por relação com outras restrições) e quais as variáveis que ultrapassaram os seus limites (*lower bounds* e *upper bounds*), se existirem (Figura D.8, da linha 454 à 464).

Durante otimizações preliminares do problema previamente descrito verificou-se que o estado INF_OR_UNBD (*Infeasable or Unbounded*) ocorria com alguma frequência. Isto deve-se à utilização de *Dual Reductions* (parâmetro *DualReductions*= 1, por definição) durante a rotina de *root presolve* (Achterberg et al., 2020, p. 20). Para saber em concreto se o problema é *Infeasable* ou *Unbounded*, é recomendada a desativação do parâmetro para que o problema não considere esse tipo de reduções de variáveis (Figura D.2, linha 29).

Capítulo 5

Implementação da Metodologia

A implementação do modelo de Calendarização Robusta proposto através de Programação Linear Inteira Mista recorre, a conjuntos distintos de n tarefas, de forma a confirmar se os objetivos e pressupostos do projeto e do modelo são satisfeitos (Capítulo 4). Nesse sentido, é necessário definir os parâmetros do problema *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP) para ajustar o modelo ás particularidades da linha de montagem que se pretende abastecer.

Após esta inicialização, utiliza-se o solver Gurobi Optimizer (GRB) para testar os vários conjuntos de tarefas e perceber, entre iterações do modelo, quais os problemas identificados na solução ótima encontrada (caso exista) e como o modelo pode ser alterado para se obter um planeamento robusto que satisfaça as necessidades do sistema e os objetivos da metodologia proposta.

O modelo de simulação construído por Almeida (2022) foi validado no contexto da movimentação dos componentes Aro da Cava da Roda do lado esquerdo (ACR-LH), Tampas e Filtros (TF) e Módulos do Motor (MM). Portanto será esse o modelo a utilizar para a recolha que dados durante a implementação da metodologia proposta e serão usados apenas os tipos de tarefas associados a estes componentes. Excluem-se os tipos de tarefa associados ao componente Aro da Cava da Roda do lado direito (ACR-RH).

Para referência ao longo do presente capítulo, a implementação do solver GRB para a otimização do modelo MILP proposto é feita no *script python* no Apêndice D.

5.1 Definição de Parâmetros

O problema MILP proposto é diretamente afetado pelo sistema em que se insere, através dos parâmetros utilizados. Estão relacionados com as características dos robôs e do tipo de tarefas, este último com grande influência na duração das tarefas e, por consequência, na performance de cada planeamento.

5.1.1 Parâmetros relacionados com o tipo de tarefa

Consoante o tipo de tarefa de cada *rack* (tt_j), o tipo de *rack* que é transportada é diferente, variando também a necessidade de *delay* (dl_j) durante a execução de uma tarefa (Tabela 5.1).

Tabela 5.1: Parâmetros associados aos tipos de *rack*. A tarefa do tipo 0 indica apenas que o robô se encontra no parque de estacionamento, no primeiro planeamento realizado.

Tipo (tt_j)	**Tipo de *rack***	**Delay (dl_j)**
0	-	-
1	Cheia	0
2	Cheia	0
3	Vazia	1
4	Vazia	1
5	Cheia	0
6	Cheia	0
7	Vazia	1

5.1.2 Parâmetros relativos ao tempo de *load* e *unload* dos robôs

Os tempos de *load* e *unload* recolhidos foram analisados no Capítulo 3. Concluiu-se que a distribuição probabilística que melhor se ajusta aos dados observados é uma Mistura Gaussiana com 2 componentes (parâmetros da distribuição na Tabela 3.6). Dado que as bibliotecas *Python* "Scipy"e "Scikit-learn"não suportam a geração de amostras de *Gaussian Mixture Model* ou Modelo de Mistura Gaussiana (GMM) sem primeiro realizar novamente o ajuste da distribuição aos dados, utilizou-se a amostra gerada aquando do primeiro ajustamento (com 1000 elementos) para criar uma Distribuição Empírica (Figura D.15). Sempre que é feito um planeamento, é gerado um valor diferente para cada parâmetro t_{ij}^{load} e t_{ij}^{unload} e para cada combinação dos índices i e j.

5.1.3 Parâmetros relativos à Deslocação dos robôs

Os parâmetros de tempo de movimentação até ao ponto de carga e descarga de cada tarefa (t_{lj}^{c} e t_{j}^{d}) são obtidos através do modelo de simulação já validado, no software *Flexsim*. Através da ferramenta *Statistics Collector* do simulador, é possível guardar a posição de início e fim de uma tarefa, o tempo que um robô demora entre o início e o ponto de carga, o tempo que demora entre o ponto de carga (após o *load* da *rack*) e o ponto de descarga e, por fim, a posição onde descarregou a *rack* e terminou a tarefa.

Para recolher todas as combinações de tipos de tarefas possíveis, foram idealizados 4 cenários com diferentes combinações de componentes a abastecer (Tabela 5.2). Isto garante que qualquer tipo de uma tarefa l ocorre antes de qualquer tipo de uma tarefa j. Se só for simulado o cenário 4, não ocorrem todas as combinações possíveis de tipos de tarefas, como é o caso da ocorrência do tipo 7 antes do tipo 6.

Foram realizadas 5 réplicas de simulação independentes por cenário, com 5 dias de duração cada, apenas para recolher elementos suficientes. Como o modelo já está validado, não foi necessário seguir as Etapas de Simulação discutidas por Law (2013).

Tabela 5.2: Cenários para a recolha de uma amostra de tempos de deslocação para cada combinação de tipos de tarefa (tt_l, tt_j).

Cenário	ACR-LH	TF	MM
1	0	1	1
2	1	0	1
3	1	1	0
4	1	1	1

Tendo em conta que este modelo de simulação utiliza o critério "*Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR) livre mais próximo da *rack*" a ser recolhida, é comum um robô aceitar uma tarefa enquanto se desloca para o parque de estacionamento. Nesses casos, os dados recolhidos pelo *Statistics Collector* serão eliminados, pois o planeamento não sabe, de momento, a posição prevista de cada robô a cada instante do mesmo, antes de obter uma solução ótima. Assim, um pressuposto a ter em conta é que t^c_{lj} e t^d_j consideram que o robô i inicia a tarefa j imediatamente a seguir à anterior l (o que na realidade não acontece, pois pode começar a deslocar-se para o parque e só depois se iniciar a tarefa seguinte).

Das amostras recolhidas de t^c_{lj} e t^d_j, são eliminadas aquelas nas quais existiu uma avaria do AMR durante a deslocação até ao ponto de carga ou descarga. A duração das avarias já é incorporada através dos parâmetros f^c_{ij} e f^d_{ij}.

Para registar a posição de um AMR, o *layout* da linha de montagem da Volkswagen Autoeuropa, Lda. (AE) foi dividido em quadrantes, cada um com um código que depende das coordenadas x e y do robô no momento de início e fim de uma tarefa (Figura 5.1).

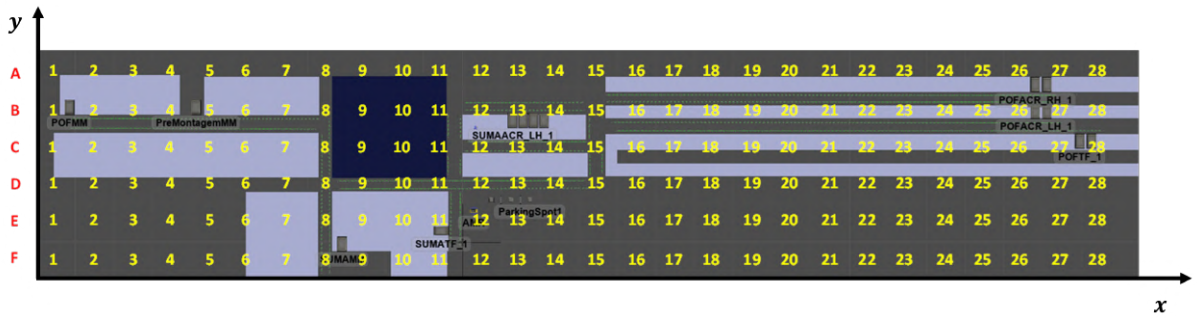


Figura 5.1: *Layout* da linha de montagem dividido em quadrantes. Cada quadrante tem 10 metros de comprimento. As letras codificam a posição no eixo y. Os números codificam a posição no eixo x.

Para apenas considerar o local de fim de uma tarefa l como o local de início da tarefa seguinte j, é necessário identificar quais os códigos associados a cada local de descarga de uma *rack* (Tabela 5.3). Depois de identificados os locais corretos e respetivos códigos, são eliminadas todas as amostras de todos os tipos de tarefas onde o local de início da tarefa não corresponde a nenhum dos códigos identificados na Tabela 5.3. São também eliminados todos os elementos cujos tempos de deslocação são nulos. Como ocorreram no final de cada réplica, o cálculo da duração da deslocação não foi terminado.

Ao saber o tipo da tarefa l (através da localização onde a tarefa j foi iniciada) e de j (tt_l e tt_j), pode calcular-se a média dos tempos recolhidos até ao ponto de carga de uma *rack*. Esta

Tabela 5.3: Código da localização de início de uma tarefa j, quando esta sucede a tarefa l, com o tipo tt_l.

Tipo da tarefa l (tt_l)	**Local de início da tarefa j**	**Código do local de início**
0	Estacionamento	D12,D13,E12,E13
1	POF ACR-LH	B26,B27
2	POF TF	C27,C28
3	SUMA ACR-LH	B13,B14
4	SUMA TF	E11
5	Pré-Montagem	B5
6	POF MM	C2
7	SUMA MM	F8,F9

é estimativa de t^c_{lj} baseada exclusivamente nos tipos das tarefas l e j. É utilizado o tempo médio, ao invés de uma distribuição probabilística, pois os tempos recolhidos para determinada combinação de tipos de tarefa (tt_l, tt_j) foram obtidos através da junção das amostras do cenário 4 com um dos restantes cenários, de forma a garantir que se obteve elementos suficientes de cada combinação. No futuro, poderá ser usada uma distribuição probabilística para determinar a duração expectável da deslocação até ao ponto de carga (t^c_{lj}) consoante a distância a que o robô i se encontra da *rack* a carregar (*load*), no momento em que a tarefa j é aceite.

$$t^c_{lj} = \begin{array}{c} \text{Tarefa } j\ (tt_j) \\ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \left[\begin{array}{cccccccc} 0 & 60.96 & 20.47 & 195.69 & 207.91 & 89.63 & 122.14 & 162.67 \\ 195.69 & 174.30 & 210.56 & 3.67 & 15.069 & 290.0 & 337.16 & 407.85 \\ 207.91 & 192.07 & 207.91 & 7.32 & 0.65 & 299.35 & 332.17 & 394.58 \\ 60.96 & 2.86 & 69.71 & 174.30 & 188.72 & 124.55 & 151.76 & 305.40 \\ 20.47 & 61.72 & 0.65 & 210.76 & 207.91 & 81.51 & 115.91 & 312.48 \\ 122.14 & 151.76 & 115.91 & 374.65 & 332.17 & 65.15 & 3.69 & 4.98 \\ 162.67 & 350.36 & 311.02 & 398.74 & 403.58 & 120.38 & 50.01 & 9.53 \\ 89.63 & 124.55 & 81.51 & 290.0 & 299.35 & 3.15 & 63.56 & 125.41 \end{array}\right] \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \end{array} \quad \text{Tarefa } l \tag{5.1}$$

Obtém-se qualquer valor da matriz do parâmetro t^c_{lj} (incluída também na Figura D.3, linhas 97 á 105) através dos tipos de tarefas, nas suas margens. No entanto, o tipo de tarefa anterior considerar depende da tarefa l:

$$\text{Tarefa } l = \begin{cases} tt_i^{last} & \text{se } l = -1 \text{ (j é a primeira tarefa),} \\ tt_l & \text{se } l \geq 0 \text{ (j não é a primeira tarefa),} \end{cases} \tag{5.2}$$

O parâmetro t^d_j apenas depende do tipo da tarefa j (Figura D.3, linha 112). Consequentemente, basta calcular a média dos tempos recolhidos entre o ponto de carga e o ponto de descarga, para todos os tipos de tarefas, como estimativa de t^d_j baseada no tipo da tarefa j. O tipo de tarefa 0 é o primeiro elemento da matriz, contudo não é usado e, por enquanto, não tem qualquer significado real no que toca ao abastecimento dos componentes.

$$t_j^d = \begin{array}{c} \text{Tarefa } j\ (tt_j) \\ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 241.09 & 177.34 & 240.36 & 89.31 & 89.63 & 42.58 & 129.12 \end{bmatrix} \end{array}$$

5.1.4 Parâmetros relativos a Avarias

Os parâmetros associados à ocorrência de avarias durante a realização de uma tarefa (f_{ij}^c e f_{ij}^d) são obtidos, como explicado no Capítulo 4, através de um modelo *Zero Inflated*, onde existe p probabilidade de ocorrência de falha durante uma tarefa. Por indicação dos responsáveis do projeto, a duração de uma avaria (tempo de não funcionamento), em minutos, pode ser modelada por uma distribuição Exponencial (parâmetros *location*= 1 e *scale*= 1.5). Utilizando a biblioteca "Scipy"é possível gerar uma amostra única e diferente para cada combinação de índices i e j, que, quando multiplicada por 0 (com $1 - p$ probabilidade) ou 1 (com p probabilidade), informa sobre a ocorrência de falha na tarefa j, quando alocada ao robô i (Figura D.3, linhas 115 á 123). Durante o teste do modelo através de vários conjuntos de tarefas, considera-se $p = 50\%$, de forma a gerar mais planeamentos em que ocorram avarias. À semelhança do parâmetro t_{lj}^c, no futuro poderá ser usada uma distribuição probabilística para determinar a probabilidade p de ocorrência de uma falha durante uma tarefa, tendo em conta o *Mean Time Between Failure* de cada robô i.

5.1.5 Outros Parâmetros

Os parâmetros "Número de tarefas"(n) e "Número de AMRs"(m) variam de cenário para cenário, pelo que serão ajustados ao longo das iterações. Para além do parâmetro tt_i^{last}, o parâmetro $time_i$ depende igualmente do último planeamento. Para testar se este parâmetro está a influenciar a alocação das tarefas, será alterado à medida que for necessário. O parâmetro Q, por depender dos parâmetros dd_j e $time_i$, será igualmente calculado à medida que se alteram os cujos de tarefas, durante o teste do modelo.

5.2 Aplicação ao Projeto AGiLE

Para testar a resolução do problema MILP em questão, utilizando o solver GRB, são feitas iterações deste mesmo problema. São testados vários conjuntos de tarefas, com parâmetros diferentes, combinados com a variação do número de AMRs.

Os conjuntos de n tarefas fornecem informação sobre os parâmetros tt_j, dl_j e dd_j, para além de um identificador da tarefa (*task_id*), para efeitos de rastreabilidade das tarefas antes e após encontrar a solução ótima, se esta existir. Os parâmetros tt_i^{last} e $time_i$ são fornecidos pelo utilizador (ou software). Para facilitar a análise dos resultados, no *script python*, tt_i^{last} será uma lista de m zeros, indicando que todos os robôs terminaram o último planeamento no parque de estacionamento, com uma tarefa do tipo 0 ($tt_i^{last} = 0,\ \forall\ i = \{0, ..., m-1\}$). Da mesma forma, $time_i$ corresponde a um lista com m elementos, onde $tt_i^{last} = 100,\ \forall\ i = \{0, ..., m-1\}$.

Para visualizar a solução ótima, quando atingida, é feito um *plot* do planeamento através do *script python* nas Figuras D.13 e D.14, bem como do valor das variáveis e parâmetros que lhe deram origem (usando as funções "*array*"). Quando não é encontrada uma solução, ou seja, o

modelo é considerado *infeasable*, são impressas na consola do IDE todas as restrições que não estão a ser cumpridas.

Na tabela 5.4 estão compilados 3 conjuntos de tarefas distintas, com diferentes n e parâmetros.

Tabela 5.4: Conjuntos de tarefas exemplo, para teste da metodologia. Cada conjunto é acompanhado pelos respetivos identificadores das tarefa (task_id), tipos de tarefa (tt_j), *delay* (dl_j) e *deadline* (dd_j). Estes conjuntos podem ser alterados a qualquer momento, se necessário, para explorar mais opções.

Conjuntos de tarefas para teste				
Conjunto	**task_id**	tt_j	dl_j	dd_j**(s)**
(1) $n = 1$	1	6	0	900
(2) $n = 2$	1	6	0	900
	2	2	0	1500
(3) $n = 3$	1	6	0	900
	2	5	0	850
	3	7	0	1050

Os *arrays*, onde são mostrados os valores finais de cada variável e os parâmetros associados, são estruturas de dados acessíveis através dos índices i, j ou i, l, j (Figura 5.2). Estes facilitam a visualização das variáveis, que de outra forma apareceriam em lista, dificultando a comparação de variáveis durante a implementação. Existem ainda listas que são acessíveis através do índice j, pois cada elemento representa uma variável ou parâmetro associado à tarefa j. Através dos *arrays* associados ás variáveis x_{ij} e y_{ilj}, uma tarefa é alocada quando $x_{ij} = 1$ e $y_{ilj} = 1$ para determinado conjunto de índices i, l, j.

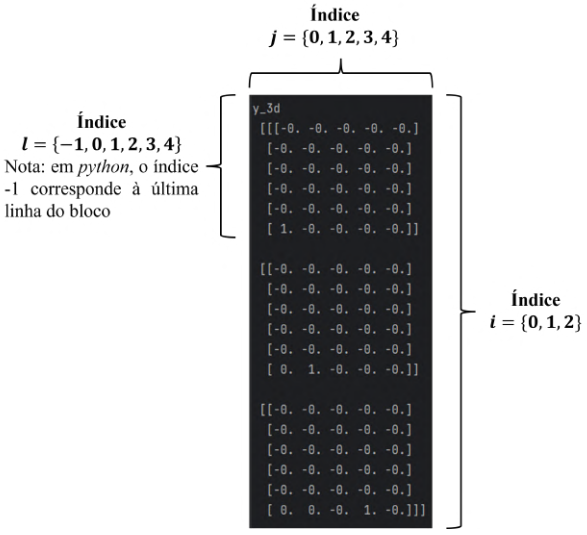


Figura 5.2: Exemplo de *array* com 3 dimensões e respetivos índices (exemplo com $n = 5$ e $m = 3$). Os *arrays* com 2 dimensões têm uma estrutura semelhante e são acessíveis através de 2 índices.

Após estas considerações, é iniciado o processo iterativo de desenvolvimento do modelo proposto. Sempre que um conjunto de tarefas for alocado a m AMRs, serão dados exemplos de planeamento após a alocação.

5.2.1 Iteração 1: Problema Original

Na Iteração 1 (Figura 5.3), é testado o problema MILP original, iniciando o processo iterativo de "Demonstração" e "Avaliação" e novo "Desenvolvimento do Artefacto" da *Design Science Research* (DSR). A descrição pormenorizada da Iteração 1 é feita na Secção E.1 em apêndice.

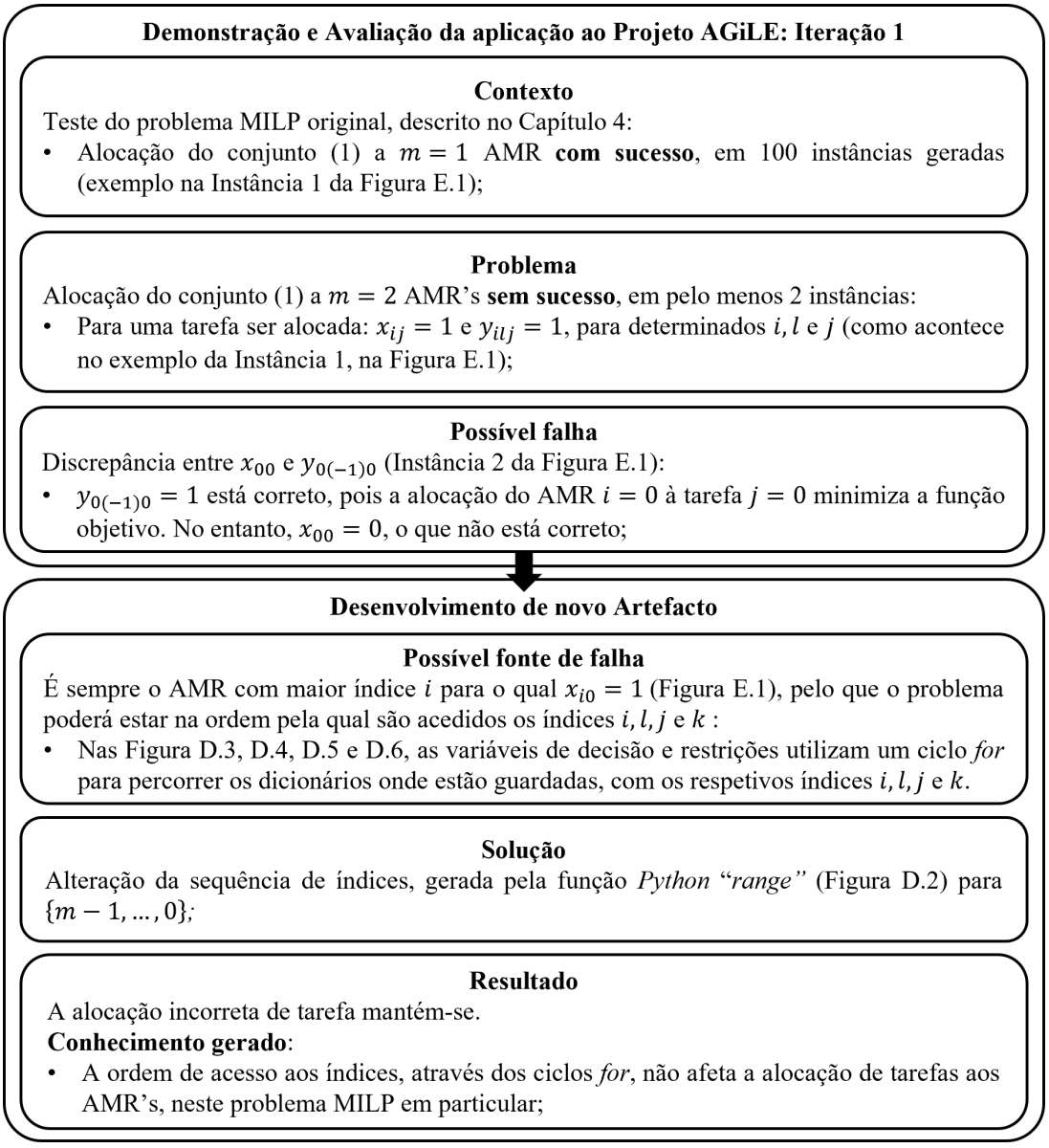
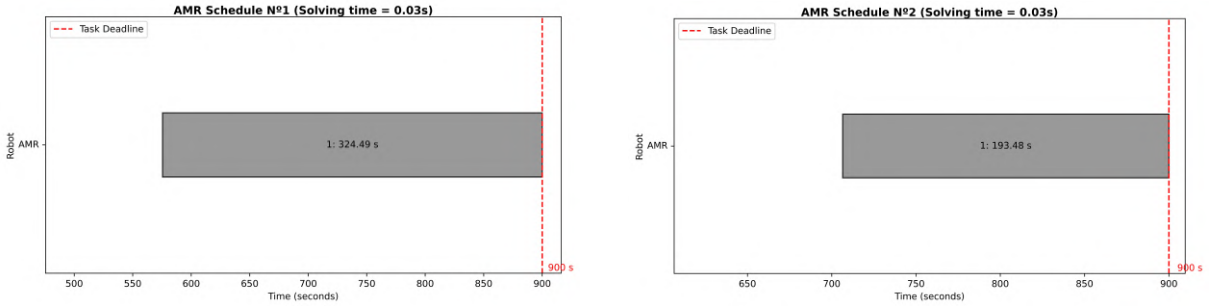


Figura 5.3: Resultados obtidos na Iteração 1: Problema Original

A alocação correta do conjunto (1) a $m = 1$ AMRs permitiu gerar todos os 100 planeamentos pedidos, cujos alguns exemplos estão representados na Figura 5.4.



(a) Conjunto (1) com avaria

(b) Conjunto (1) sem avaria

Figura 5.4: Exemplos de planeamentos gerados pela alocação do conjunto (1) a $m = 1$ robôs (Iteração 1).

5.2.2 Iteração 2: Restrições de Sequenciamento que dependem de n e m

Dada a alocação incorreta de tarefas na Iteração 1, mesmo após clarificada a ordem dos elementos de cada conjunto de índices, a Iteração 2 inicia-se com o mesmo problema da Iteração 1, mas introduz uma nova solução, explicada na Figura 5.5. A descrição pormenorizada da Iteração 2 é feita na Secção E.2 em apêndice.

Como resultado desta iteração, a Equação (5.3) é adicionada ao conjunto de Restrições de Sequenciamento e são alterados os parâmetros *IntFeasTol*, *IntegralityFocus* e *Aggregate*, o que leva à alocação do conjunto (1) a $m = 1$, $m = 2$ e $m = 3$ AMRs, exemplificada na Figura 5.6.

$$F_{ilj} - \text{time}_i + Q \times (1 - x_{ij} + y_{ilj}) \geq p_{ilj}, \qquad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ (n \leq m) \\ (l = -1) \end{array} \tag{5.3}$$

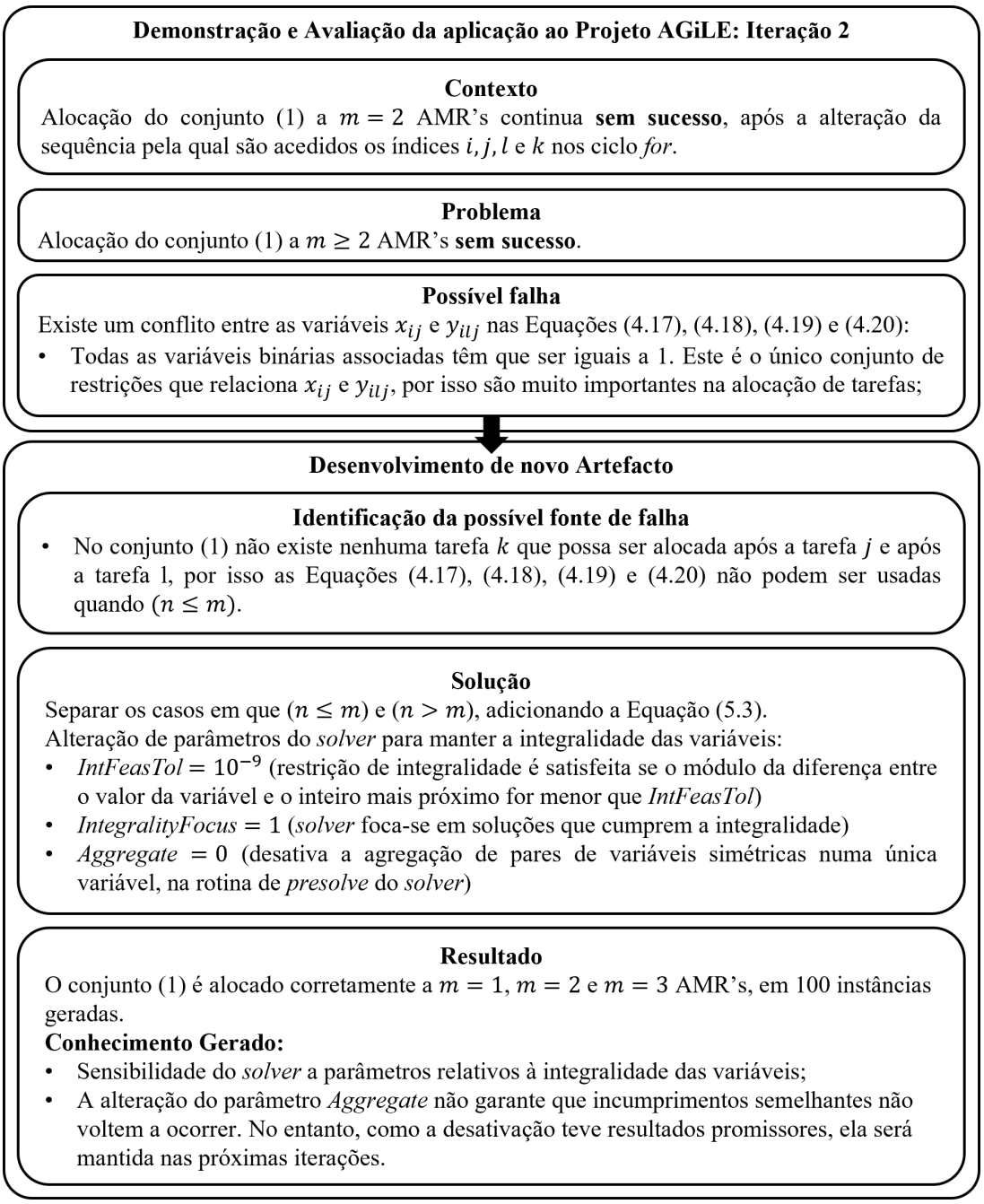
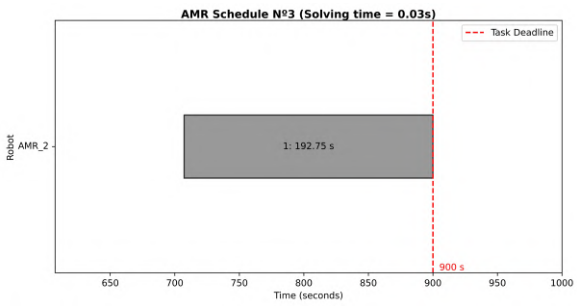
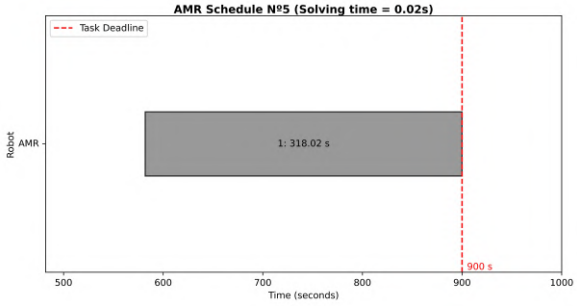


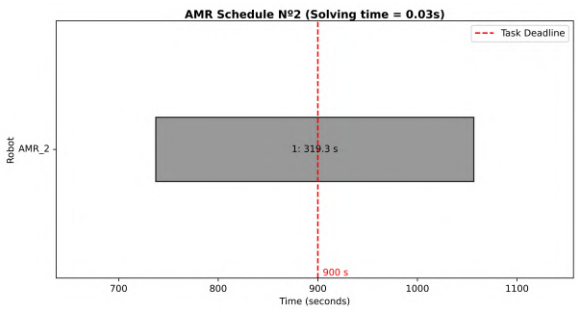
Figura 5.5: Resultados obtidos na Iteração 2: Restrições de Sequenciamento que dependem de *n* e *m*



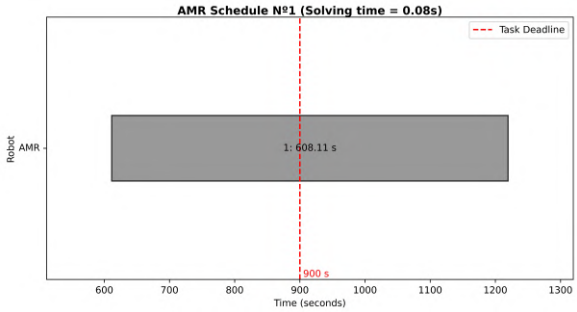
(a) Conjunto (1) sem avaria



(b) Conjunto (1) com avaria



(c) Conjunto (1) sem avaria e com *delay*



(d) Conjunto (1) com avaria e com *delay*

Figura 5.6: Exemplos de planeamentos gerados pela alocação do conjunto (1) a $m = 2$ robôs (Iteração 2)

5.2.3 Iteração 3: Alocação de tarefas após a primeira tarefa de cada robô

Após testar a otimização do problema de alocação de um conjunto com 1 tarefa, é necessário aumentar o número de tarefas para testar as Restrições de Sequenciamento das Equações (4.17), (4.18), (4.19), (4.20) e (5.3). Este teste, que dá lugar à Iteração 3, está descrito na Figura 5.7. A descrição da Iteração 3 é feita na Secção E.3 em apêndice.

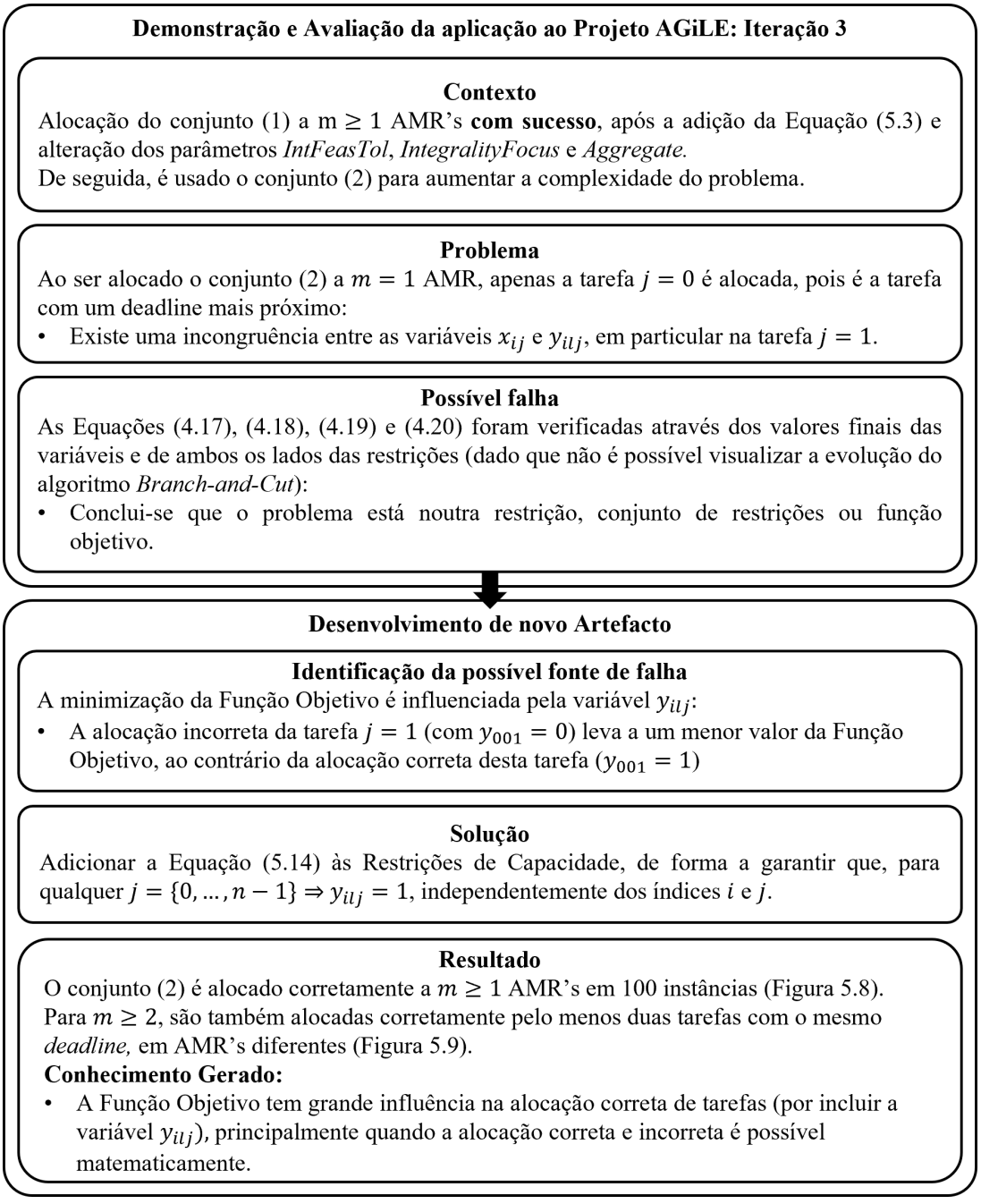
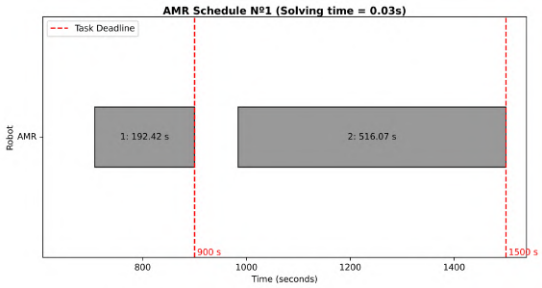


Figura 5.7: Resultados obtidos na Iteração 3: Alocação de tarefas após a primeira tarefa de cada robô

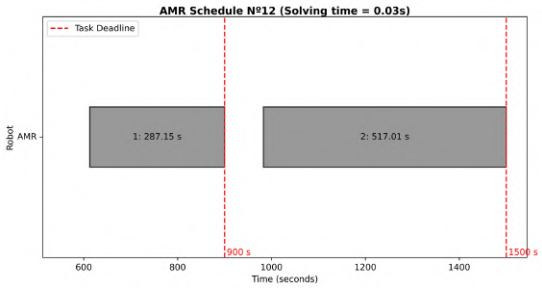
A Iteração 3 responde ao problema da alocação incorreta do conjunto (2) a m AMRs. Para o resolver, é adicionada a Equação 5.4 às Restrições de Capacidade já existentes. Na Figura 5.9 estão representados alguns exemplos da alocação correta do conjunto (2) a $m = 2$ AMRs.

$$\sum_{i=0}^{m-1}\sum_{l=-1}^{n-1} y_{ilj} = 1, \qquad \begin{array}{c}\forall\, j = \{0,\ldots,n-1\}\\ (l \neq j)\\ (n > m)\end{array} \tag{5.4}$$

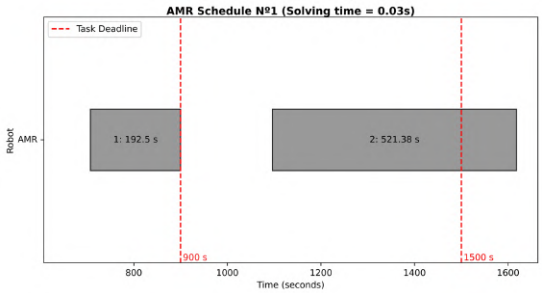
A adição desta restrição permite a alocação do conjunto (2) a $m \geq 1$ em todos os 100 planeamentos gerados. Exemplos destas alocações estão representados nas Figuras 5.8, 5.9 e 5.10. Inclusive, quando as tarefas têm deadlines simultâneos, a alocação do conjunto (2) é feita corretamente em todos os 100 planeamentos gerados (Figura 5.11).



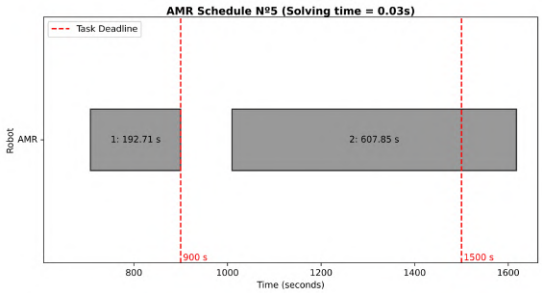
(a) Conjunto (2) sem avaria



(b) Conjunto (2) com avaria na tarefa com *task_ID*= 1

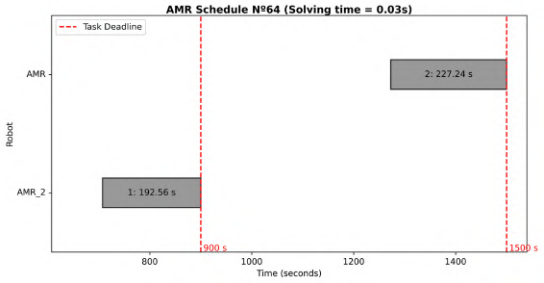


(c) Conjunto (2) sem avaria e tarefa com *task_ID*= 2 tem delay

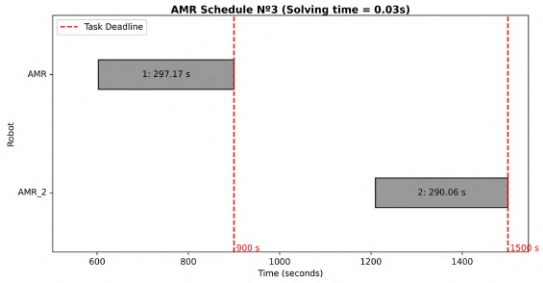


(d) Conjunto (2) com avaria e tarefa com *task_ID*= 2 tem delay

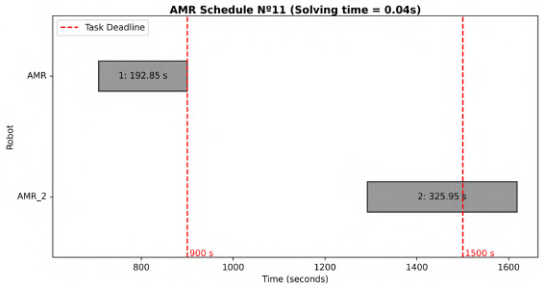
Figura 5.8: Exemplos de planeamentos gerados pela alocação do conjunto (2) a $m = 1$ robôs (Iteração 3). As tarefas com *delay* surgem por alteração da tarefa com *task_ID*= 2, onde o tipo da tarefa foi alterado de $tt_1 = 2$ para $tt_1 = 4$.



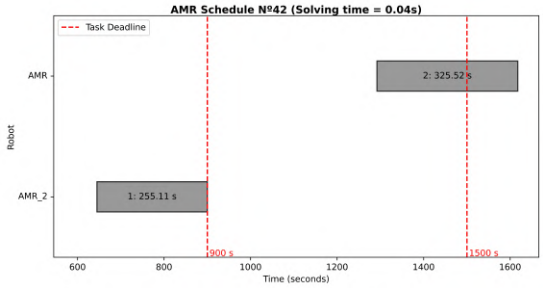
(a) Conjunto (2) sem avaria



(b) Conjunto (2) com avaria na tarefa com *task_ID*= 1

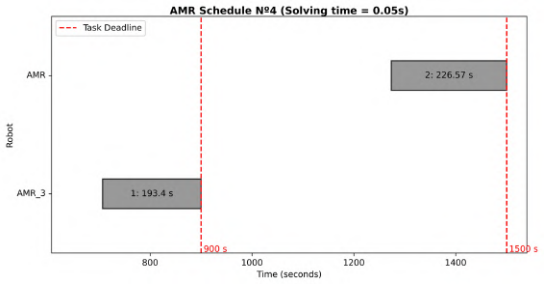


(c) Conjunto (2) sem avaria e tarefa com *task_ID*= 2 tem delay

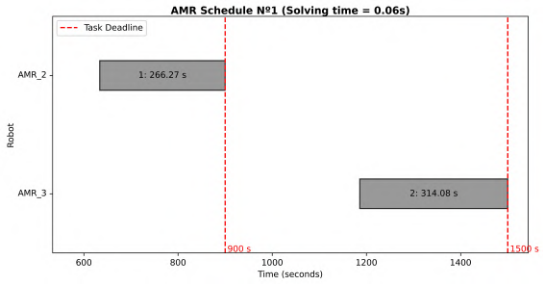


(d) Conjunto (2) com avaria e tarefa com *task_ID*= 2 tem delay

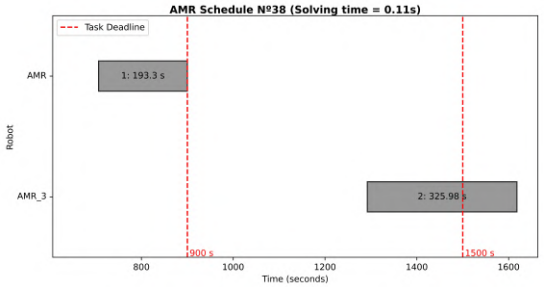
Figura 5.9: Exemplos de planeamentos gerados pela alocação do conjunto (2) a $m = 2$ robôs (Iteração 3). As tarefas com *delay* surgem por alteração da tarefa com *task_ID*= 2, onde o tipo da tarefa foi alterado de $tt_1 = 2$ para $tt_1 = 4$.



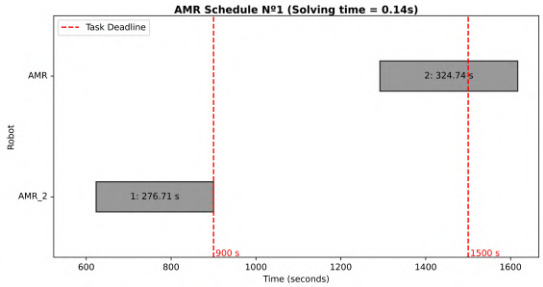
(a) Conjunto (2) sem avaria



(b) Conjunto (2) com avaria na tarefa com *task_ID*= 1

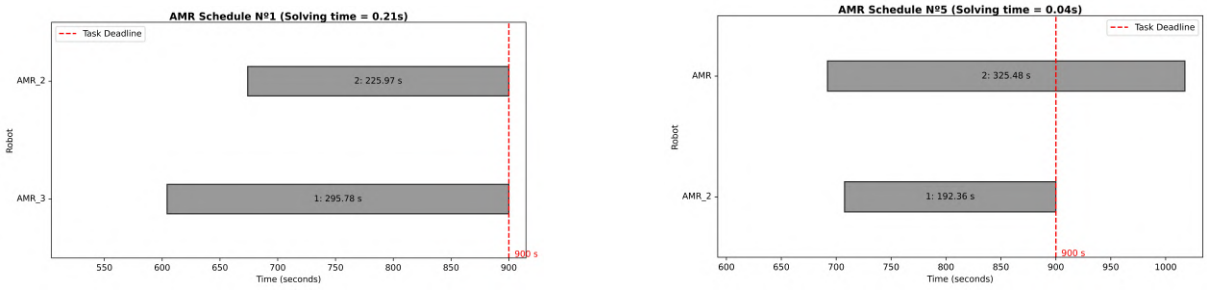


(c) Conjunto (2) sem avaria e tarefa com *task_ID*= 2 tem delay



(d) Conjunto (2) com avaria e tarefa com *task_ID*= 2 tem delay

Figura 5.10: Exemplos de planeamentos gerados pela alocação do conjunto (2) a $m = 3$ robôs (Iteração 3). As tarefas com *delay* surgem por alteração da tarefa com *task_ID*= 2, onde o tipo da tarefa foi alterado de $tt_1 = 2$ para $tt_1 = 4$.



(a) Alocação do conjunto (2) sem *delay*.

(b) Alocação do conjunto (2) com *delay*

Figura 5.11: Exemplos de planeamentos gerados pela alocação do conjunto (2) a $m = 3$ robôs, com *deadlines* simultâneos (Iteração 3).

5.2.4 Iteração 4: Técnica de Reformulação-Linearização (RLT)

Após a alocação correta do conjunto (2) a $m \geq 1$ robôs, segue-se a alocação do conjunto (3) a $m \geq 1$ AMRs. Deste teste surge a Iteração 4 (Figura 5.12), que se foca no uso da *Reformulation-Linearization Technique* ou Técnica de Reformulação-Linearização (RLT) para resolver os problemas que surgem desta alocação. A descrição pormenorizada da Iteração 4 é feita na Secção E.4 em apêndice.

A RLT é uma técnica de linearização de problemas de Programação Inteira Mista (linear ou polinomial). É considerada uma técnica robusta para a representação linear da multiplicação de duas variáveis, neste caso, $x_{ij} \times y_{ilj}$.

Para o efeito, é necessário adicionar as Equações (5.5), (5.6), (5.7) e (5.8) e substituir os termos $(4 - x_{ij} - x_{ik} - y_{ilj} - y_{ijk})$ e $(3 - x_{ij} - x_{ik} - y_{ilj} + y_{ijk})$ nas Equações (4.17), (4.18), (4.19) e (4.20) pelos termos $(2 - z_{ilj} - z_{ijk})$ e $(1 - z_{ilj} + z_{ijk})$, respetivamente.

$$z_{ilj} \leq x_{ij} \qquad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ (l \neq j) \end{array} \tag{5.5}$$

$$z_{ilj} \leq y_{ij} \qquad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ (l \neq j) \end{array} \tag{5.6}$$

$$z_{ilj} \geq x_{ij} + y_{ij} - 1 \qquad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ (l \neq j) \end{array} \tag{5.7}$$

$$z_{ilj} \in \{0, 1\}, \qquad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ (l \neq j) \end{array} \tag{5.8}$$

De forma a garantir que, para cada tarefa j, existe uma e uma só variável z_{ilj} tal que $z_{ilj} = 1$, a Equação (5.4) é substituída pela Equação (5.9).

$$\sum_{i=0}^{m-1} \sum_{l=-1}^{n-1} z_{ilj} = 1, \qquad \begin{array}{c} \forall\, j = \{0, \ldots, n-1\} \\ (l \neq j) \end{array} \tag{5.9}$$

No entanto, como mencionado na Figura 5.12, estas alterações não evitam a sobreposição de tarefas no mesmo AMR, como exemplificado na Figura 5.13.

Demonstração e Avaliação da aplicação ao Projeto AGiLE: Iteração 4

Contexto

O conjunto (2) foi alocado corretamente a $m \geq 1$ AMR's em 100 instâncias geradas. Para isto, foi adicionada a Equação (5.4) ao conjunto de Restrições de Capacidade.

Problema

É testada a alocação o conjunto (3) a $m \geq 1$:

- Se $m = 1$ torna problema *infeasible* ou inválido, pois existem deadlines simultâneos;
- Se $m = 2$, algumas alocações são bem sucedidas (as tarefas $j = 1$ e $j = 2$ são alocadas no mesmo robô e $j = 0$ noutro robô). No entanto, algumas instâncias mostram a sobreposição das tarefas $j = 0$ e $j = 2$ no mesmo robô, o que implica novamente incongruências entre as variáveis x_{ij} e y_{ilj} (Figura E.8);

Possível falha

A Função Objetivo foi novamente verificada através dos valores finais das variáveis e de ambos os lados das restrições:

- A minimização da Função Objetivo leva à alocação incorreta das tarefas $j = 0$ e $j = 2$ no mesmo robô, embora a solução considerada "correta" também seja possível;

Desenvolvimento de novo Artefacto

Identificação da possível fonte de falha

As Restrições de Sequenciamento para $n > m$ não estão a impor os valores das variáveis x_{ij} e y_{ilj}:

- Qualquer combinação destas variáveis não infringe as restrições, o que significa que não são suficientemente rígidas para obter os valores de x_{ij} e y_{ilj}esperados;

Solução

Introdução da Técnica de Reformulação-Linearização (RLT), linearizando a expressão $x_{ij} \times y_{ilj}$ e adicionando as restrições das Equações (5.5), (5.6), (5.7) e (5.8).

A Restrição de Capacidade da Equação (5.4) é também substituída pela Equação (5.9), garantindo que, para qualquer $j = \{0, \dots, n-1\} \Rightarrow z_{ilj} = 1$, independentemente dos índices i e j.

Substituem-se também os termos $(4 - x_{ij} - x_{ik} - y_{ilj} - y_{ijk})$ e $(3 - x_{ij} - x_{ik} - y_{ilj} + y_{ijk})$ nas Equações (4.17), (4.18), (4.19) e (4.20) pelos termos $(2 - z_{ilj} - z_{ijk})$ e $(1 - z\_ilj + z\_ijk)$, respetivamente.

Resultado

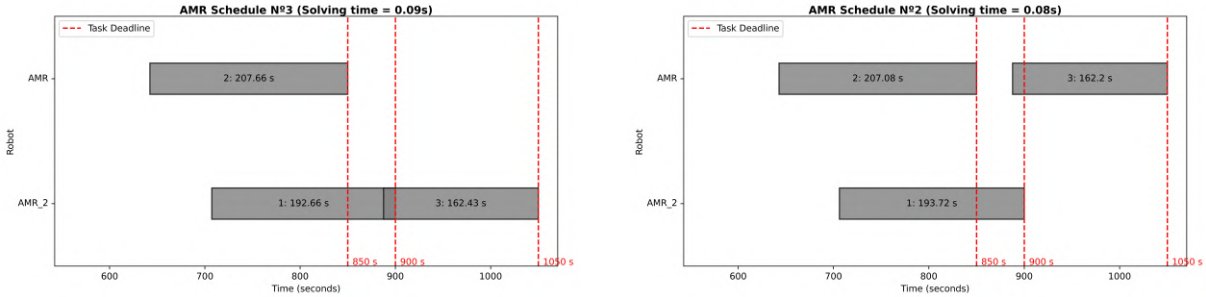
Em 10 instâncias geradas, existem ainda algumas onde a alocação de tarefas é incorreta, com sobreposição das tarefas j=1 e j=2 (Figura 5.11):

- A Função Objetivo é novamente responsável pela escolha da sequência errada de tarefas (e onde o Cenário 5, na Tabela E.1, é o cenário correto);
- Através da representação linear de cada restrição e a diferença entre o valor do lado esquerdo e direito , dada pelo *solver* (Figura E.9)

Conhecimento Gerado

- As Restrições de Sequenciamento não estão a impedir a sobreposição de tarefas. Apenas garantem que a soma da duração das tarefas alocadas a cada robô consegue ser inserida no intervalo de tempo previsto (Q) (como provado no Apêndice F).

Figura 5.12: Resultados obtidos na Iteração 4: Implementação da RLT



(a) Alocação incorreta, com sobreposição.

(b) Alocação correta, sem sobreposição

Figura 5.13: Exemplos de planeamentos gerados pela alocação do conjunto (3) a $m = 2$ robôs (Iteração 4).

5.2.5 Iteração 5: Outras Restrições de Sequenciamento

Dado que as Restrições de Sequenciamento não são capazes de garantir que não existe sobreposição entre duas ou mais tarefas no mesmo robô, é necessário outro tipo de restrições que substitua o conjunto de Restrições de Sequenciamento atuais. Além das variações das Restrições de Sequenciamento mencionadas na Secção 5.2.2, após a implementação da variável z, foram testadas a restrição da Equação (5.10), juntamente com as Equações (4.17), (4.18), (4.19) e (4.20), sem sucesso na alocação em todos os planeamento. A Equação (5.11) representa outra restrição adicionada ás Restrições de Sequenciamento, após a adição da variável z, mas que também não teve sucesso na alocação de tarefas.

$$\mathrm{F_{ijk}} - \mathrm{S_{ilj}} + \mathrm{Q} \times (2 - \mathrm{z_{ilj}} - \mathrm{z_{ijk}}) \geq \mathrm{p_{ijk}} + \mathrm{p_{ilj}}, \quad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ \forall\, k = \{0, \ldots, n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{5.10}$$

$$\mathrm{S_{ijk}} - \mathrm{F_{ilj}} + \mathrm{Q} \times (2 - \mathrm{z_{ilj}} - \mathrm{z_{ijk}}) \geq 0, \quad \begin{array}{c} \forall\, i = \{0, \ldots, m-1\} \\ \forall\, l = \{-1, \ldots, n-1\} \\ \forall\, j = \{0, \ldots, n-1\} \\ \forall\, k = \{0, \ldots, n-1\} \\ (l \neq j) \bigwedge (j \neq k) \bigwedge (k \neq l) \end{array} \tag{5.11}$$

As Equações 5.10 e 5.11 pretendem eliminar a sobreposição entre tarefas exemplificada na Figura 5.14, ao controlar o intervalo de tempo entre os instantes F_{ijk} e S_{ilj} para que seja maior ou igual à soma da duração das tarefas ($p_{ijk} + p_{ilj}$), ou ao controlar o intervalo de tempo entre os instantes S_{ijk} e F_{ilj} para que seja maior ou igual a zero, quando necessário.

Qualquer restrição deste tipo irá resultar na alocação incorreta de alguns planeamentos, pois a função objetivo irá escolher a sequência de tarefa que efetivamente minimize o seu valor. Quando $z_{ilj} = z_{ijk} = 1$, estas restrições ou as restrições iniciais de sequenciamento cumprem a sua função. Quando uma das variáveis binárias é nula, o lado direito destas restrições passa a ser maior ou igual a zero ou maior ou igual a um número negativo, o que permite que a diferença $F_{ijk} - S_{ilj}$, $S_{ijk} - F_{ilj}$, $F_{ijk} - F_{ilj}$ ou $S_{ijk} - S_{ilj}$ tome qualquer número real. Assim, independentemente

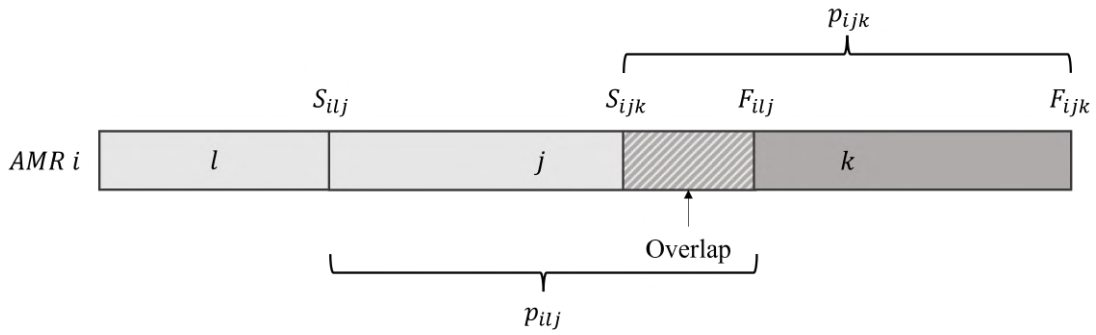


Figura 5.14: Exemplo da sobreposição de duas tarefas j e l, quando alocadas ao mesmo robô i.

de z_{ilj}, z_{ijk}, x_{ij} e y_{ilj}, é sempre possível alocar as tarefas com a menor duração, mesmo havendo sobreposição.

Restrições de Sequenciamento em que as tarefas dependem de tarefas anteriores são geralmente formuladas através de restrições do tipo Big-M. Quando introduzidas as *keywords "sequence-dependent task duration"* + "MILP" no motor de buscar *Google Schoolar* (por ordem decrescente de relevância), é possível encontrar várias publicações onde, através de restrições semelhantes, a alocação de tarefas cujas durações dependem da tarefa anterior é feita sem sobreposição. Para além do problema no qual a presente metodologia é baseada (Ghaleb & Taghipour, 2023), Srikar e Ghosh (1986, p. 1464), Meng, Gao et al. (2022), Kelly e Zyngier (2007), Meng, Zhang et al. (2022, pp. 8, 9), Abreu et al. (2023, p. 4) e Yalaoui e Quy Nguyen (2021, p. 17) usam *Big-M constraints* para sequenciar tarefas num mesmo recurso. Uma característica comum a todos estes problemas é a utilização dos tempos de conclusão de cada tarefa (*completion time*) para delimitar o fim de uma tarefa e o início da tarefa seguinte nesse mesmo instante. O *completion time* de uma tarefa j é calculado pelo somatório da duração das tarefas alocadas ao mesmo recurso até à tarefa j, portanto basta garantir que a diferença entre o *completion time* de duas tarefas sequenciais j e k (por esta ordem) é maior ou igual à duração da tarefa j (exemplificado no Apêndice F).

Kelly e Zyngier (2007), Yalaoui e Quy Nguyen (2021, p. 20) usam o *completion time* e comparam os *time slots* discretos onde cada tarefa termina, de forma a garantir que não existem sobreposições (ao contrário da variável *completion time* contínua habitualmente utilizada).

Todas as publicações mencionadas anteriormente utilizam apenas o instante de conclusão da tarefa para garantir que não existem sobreposições, juntamente com o somatório das durações da tarefas para calcular o *completion time* de uma tarefa. Já na presente metodologia, o modelo matemático necessita de limitar igualmente o instante de início de cada tarefa, o que torna mais difícil a extrapolação deste tipo de restrições para este problema em particular.

Durante o desenvolvimento da presente dissertação, foram contactados os autores de Ghaleb e Taghipour (2023) sobre a possibilidade de adaptação das restrições das Equações F.1 e F.2 ao problema MILP formulado aquando do projeto AGiLE, face aos obstáculos descritos neste capítulo, mas até à data não se obteve qualquer resposta.

Capítulo 6

Conclusões

Esta dissertação apresenta o AGiLE como um projeto de inovação, destinado a revolucionar a intralogística em linhas de produção e montagem pela integração de *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR)s e periféricos sensorizados (*racks*) numa arquitetura em rede descentralizada. Pretende-se substituir a alocação de tarefas puramente reativa por um modelo de Calendarização Robusta, através da otimização de um problema de Programação Linear Inteira Mista (MILP), minimizando a duração das tarefas a desempenhar pelos robôs. Os problemas *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP), acompanhados por um *solver* como o Gurobi (Gurobi Optimizer (GRB)), são uma ferramenta versátil aquando da Calendarização Robusta, oferecendo a flexibilidade necessária a sistemas estocásticos e a adaptabilidade da solução aos processos de qualquer linha de produção ou montagem. Para o efeito, a formulação matemática do problema é semelhante à utilizada por Ghaleb e Taghipour (2023), adotando a metodologia de investigação *Design Science Research*, que proporciona uma abordagem estruturada para o desenvolvimento de novas tecnologias.

Primeiramente, são discutidas sugestões de melhoria do modelo de simulação do abastecimento através do AGiLE, desenvolvido por Almeida (2022) durante a implementação do projeto na Volkswagen Autoeuropa, Lda. (AE) e onde o autor optou pela Simulação de Eventos Discretos (DES), através do *software Flexsim*. Estas sugestões deverão, no futuro, ser aplicadas ao modelo já validado por Almeida (2022) e simuladas segundo as etapas descritas por Law (2013), de forma a poder inferir sobre a performance das melhorias sugeridas.

O problema de Calendarização Robusta abordado pode ser considerado um problema de *Unrelated Parallel Machines* (UPMP), que envolve o abastecimento de componentes por AMRs a uma linha de montagem, tendo em consideração várias fontes de incerteza, como avarias durante a execução das tarefas e localização dos robôs no momento em que uma tarefa é aceite. Pretende-se construir um planeamento das tarefas a realizar por cada robô, gerando uma Calendarização de Alocação Estática. Na notação introduzida por Graham et al. (1979), o problema pode ser classificado como ($R \mid resources \mid$"*Duração das Tarefas*"). A formulação matemática do problema MILP pressupõe a minimização da função objetivo "*Duração das Tarefas*", segundo os parâmetros

e variáveis de decisão adequados ao tipo de tarefas e juntamente com um conjunto de restrições de Capacidade, Precedência, Sequenciamento e Integralidade. O problema é depois otimizado pelo *solver* GRB, através do algoritmo *Branch-and-Cut*.

Durante a implementação do modelo foram realizadas iterações com diferentes conjuntos de tarefas e número de AMRs livres, para perceber se os objetivos de alocação de tarefas a robôs são cumpridos. Se existir algum pressuposto que não esteja de acordo com o esperado, é identificado o problema e proposta uma solução.

Na Iteração 1 é testado o problema MILP original, descrito no Capítulo 4. Embora a alocação de $n = 1$ tarefas a $m = 1$ robôs tenha tido sucesso, quando gerados 100 calendários, o mesmo não acontece quando existem mais robôs disponíveis. Foi então testada e posteriormente descartada a hipótese da ordem pela qual são acedidos os índices nos dicionários de variáveis de decisão e de restrições ser a fonte deste problema.

A Iteração 2 foca-se em corrigir a falha na alocação de tarefas a múltiplos AMRs identificada na iteração anterior. A análise do problema revelou que as equações do modelo MILP geram um conflito quando o número de AMRs era maior ou igual ao número de tarefas. Para solucionar essa falha, foi adicionada uma nova equação e os parâmetros do solver foram ajustados para garantir a integralidade das variáveis. Como resultado, o problema MILP foi capaz de alocar corretamente as tarefas a 1, 2 e 3 AMRs em todos os planeamentos testados. Esta iteração permitiu também conhecer mais sobre a sensibilidade do *solver* a parâmetros de integralidade e a importância de desativar o parâmetro *Aggregate* para evitar falhas semelhantes no futuro.

Parâmetros como o *IntFeasTol* e *IntegralityFocus*, bem como o parâmetro *Aggregate* não garantem ser a solução para a quebra de integralidade. Se o problema continuar a ser inviável mesmo após desligar a agregação de variáveis no *presolve*, então é necessário analisar o *output* da otimização e, em conjunto com o manual do *solver* (Gurobi Optimization, 2023a), perceber quais as alterações a fazer, se existirem. Uma das soluções apresentadas é desligar a rotina de *presolve* por completo. No entanto, isto não impede a inviabilidade do problema e apenas elimina a possibilidade de ser a rotina de *presolve* a fonte do erro. Retirar a rotina de *presolve* tem também penalizações de performance da otimização, o que, para modelos com elevados n e m, pode tornar-se um fator de risco no planeamento atempado de tarefas para os robôs.

A Iteração 3 concentrou-se em aumentar a capacidade do modelo MILP para lidar com problemas mais complexos. O conjunto com 2 tarefas, que apresenta maior dificuldade devido à presença de *deadlines* mais próximos, foi utilizado como teste. A origem da falha na alocação deste conjunto de tarefas foi identificada e uma solução eficaz foi desenvolvida. A implementação da solução resultou em um modelo MILP capaz de alocar corretamente um conjunto de 2 tarefas a múltiplos AMRs, em todos os planeamentos testados. Além disso, esta iteração permitiu compreender a influência da Função Objetivo na alocação de tarefas e a importância de considerar a complexidade do problema ao formular o modelo MILP.

Nas Iterações 4 e 5 são aplicadas outro tipo de restrições para tentar colmatar esta falha por parte das Restrições de Sequenciamento, em conjunto com a Função Objetivo. Na Iteração 4, em particular, ao tentar alocar 3 tarefas a múltiplos AMRs, percebeu-se que as Restrições de Sequenciamento utilizadas até então não estão a impor os valores corretos às variáveis do problema, embora todos os esforços até então tenham culminado na alocação de 2 tarefas a múltiplos robôs. Uma alternativa é a **Técnica de Reformulação-Linearização (RLT)**, mas que

rapidamente se revelou inútil na resolução do problema. Tanto as Restrições de Sequenciamento como a *Reformulation-Linearization Technique* ou Técnica de Reformulação-Linearização (RLT) não impedem a sobreposição de tarefas.

Pode concluir-se que, até ao momento, o **modelo de Calendarização Robusta proposto não é válido**, embora seja possível a alocação de duas tarefas a vários robôs. Para controlar o instante de fim e início das tarefas, o método **Big-M** é capaz de garantir que a sequência de tarefas pode ser concluída dentro do *makespan* previsto, mas incapaz de impedir sobreposições nas Restrições de Sequenciamento. Vários autores mostram a sua utilidade em problemas de Programação Linear, onde as restrições permitem a sequenciação de tarefas sem sobreposição. No entanto, estas apenas controlam a diferença entre os instantes de conclusão de duas tarefas diferentes, sem possibilidade de intervalo entre duas tarefas. No problema MILP proposto, é necessário limitar também o instante de início das tarefas a alocar, permitindo intervalos entre tarefas, nos quais o robô está livre.

Como trabalho futuro, de forma a colmatar esta falha na alocação de tarefas através da metodologia de Calendarização Robusta, propõe-se a substituir o problema MILP pelo algoritmo *Simulated Annealing* como passo seguinte para obter um modelo de Calendarização Robusta válido, tendo em conta os pressupostos apresentados. Se esta abordagem não resultar num modelo válido, propõe-se a utilização de *time slots* para verificar se existem tarefas que partilhem a mesma posição. A natureza discreta e fixa dos *time slots* irá, à partida, eliminar a necessidade de variáveis contínuas e, como consequência, a necessidade de utilização de restrições do tipo **Big-M**. Prevê-se, no entanto, a perda de precisão nos instantes de início e fim de uma tarefa, podendo variar-se o intervalo de tempo correspondente a um *time slot*. Após ultrapassar este obstáculo, é possível aumentar a complexidade da metodologia proposta, adicionando o cálculo dos *deadlines* através do número de componentes existentes em cada *Point of Fit* (POF) ao longo do tempo e a integração do nível de bateria para prever o instante em que cada robô precisa de voltar ao parque de estacionamento para carregar.

Em síntese, a Calendarização Robusta para a alocação de tarefas a AMRs é um processo que desvendou tanto vantagens quanto desafios. As vantagens incluem a capacidade de lidar com incertezas de complexidade variável e a flexibilidade de adaptação às necessidades da indústria. No entanto, surgiram desvantagens na forma de problemas de integralidade e sequenciamento de tarefas, que requerem uma nova estratégia para a alocação de tarefas sem sobreposições. A superação destes obstáculos impulsionam o progresso desta metodologia para além da alocação de, no máximo, 2 tarefas a múltiplos AMRs disponíveis, o que só por si já demonstra o sucesso da MILP na modelação de problemas de Calendarização Robusta.

Referências Bibliográficas

Abreu, L. R., Prata, B. A., Nagano, M. S., & Framinan, J. M. (2023). A constraint programming-based iterated greedy algorithm for the open shop with sequence-dependent processing times and makespan minimization. *Computers & Operations Research*, *160*, 106386. https://doi.org/10.1016/j.cor.2023.106386 (ver p. 92).

Achterberg, T., Bixby, R. E., Gu, Z., Rothberg, E., & Weninger, D. (2020). Presolve Reductions in Mixed Integer Programming. *INFORMS Journal on Computing*, *32*(2), 473–506. https://doi.org/10.1287/ijoc.2018.0857 (ver pp. 72, 73, 180).

Afonso, T., C. Alves, A., Carneiro, P., & Vieira, A. (2021). Simulation pulled by the need to reduce wastes and human effort in an intralogistics project. *International Journal of Industrial Engineering and Management*, *12*(4), 274–285. https://doi.org/10.24867/IJIEM-2021-4-294 (ver pp. 7, 11, 15, 18).

Almeida, M. (2022). *Estudo do Abastecimento de uma Linha de Montagem:Projeto AGILE* [Masters Dissertation]. Universidade Nova de Lisboa. (Ver pp. 2, 4, 33, 36, 39, 44, 51, 60, 72, 75, 93).

Aramesh, S., Mousavi, S., Mohagheghi, V., Zavadskas, E., & Antucheviciene, J. (2021). A soft computing approach based on critical chain for project planning and control in real-world applications with interval data. *Applied Soft Computing*, *98*, 106915. https://doi.org/10.1016/j.asoc.2020.106915 (ver pp. 12, 19).

Bekishev, Y., Pisarenko, Z., & Arkadiev, V. (2023). FMEA Model in Risk Analysis for the Implementation of AGV/AMR Robotic Technologies into the Internal Supply System of Enterprises. *Risks*, *11*(10), 172. https://doi.org/10.3390/risks11100172 (ver pp. 10, 17).

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer. (Ver pp. 42, 43).

Chakrabarti, A., & Ghosh, J. K. (2011). AIC, BIC and Recent Advances in Model Selection. Em *Philosophy of Statistics* (pp. 583–605). Elsevier. https://doi.org/10.1016/B978-0-444-51862-0.50018-6 (ver p. 43).

Chen, Y.-C. (2017). A tutorial on kernel density estimation and recent advances. *Biostatistics & Epidemiology*, *1*(1), 161–187. https://doi.org/10.1080/24709360.2017.1396742 (ver p. 41).

Chen, Y., Xu, A., He, Q.-C., & Chen, Y.-J. (2023). Smart navigation via strategic communications in a mixed autonomous paradigm. *Production and Operations Management*, poms.13987. https://doi.org/10.1111/poms.13987 (ver pp. 10, 16).

Cohen, I., Postek, K., & Shtern, S. (2023). An adaptive robust optimization model for parallel machine scheduling. *European Journal of Operational Research*, *306*(1), 83–104. https://doi.org/10.1016/j.ejor.2022.07.018 (ver pp. 11, 21–26, 31).

Cokelaer, T., Kravchenko, A., Lahdjirayhan, Msat59, Varma, A., L, B., Stringari, C. E., Brueffer, C., Broda, E., Pruesse, E., Karthikeyan Singaravelan, Zhengyi Li, Padgham, M., & Negodfre. (2023, janeiro). cokelaer/fitter: v1.5.2. https://doi.org/10.5281/ZENODO.7497983

De Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, *182*(2), 481–501. https://doi.org/10.1016/j.ejor.2006.07.009 (ver p. 36).

Dehghan-Sanej, K., Eghbali-Zarch, M., Tavakkoli-Moghaddam, R., Sajadi, S., & Sadjadi, S. (2021). Solving a new robust reverse job shop scheduling problem by meta-heuristic algorithms. *Engineering Applications of Artificial Intelligence*, *101*, 104207. https://doi.org/10.1016/j.engappai.2021.104207 (ver pp. 12, 22, 29).

Duan, J., & Wang, J. (2022). Robust scheduling for flexible machining job shop subject to machine breakdowns and new job arrivals considering system reusability and task recurrence. *Expert Systems with Applications*, *203*, 117489. https://doi.org/10.1016/j.eswa.2022.117489 (ver pp. 11, 19–21, 24, 26, 27, 29, 31).

Fragapane, G., de Koster, R., Sgarbossa, F., & Strandhagen, J. O. (2021). Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda. *European Journal of Operational Research*, *294*(2), 405–426. https://doi.org/10.1016/j.ejor.2021.01.019 (ver pp. 10, 15).

Fragapane, G., Ivanov, D., Peron, M., Sgarbossa, F., & Strandhagen, J. O. (2022). Increasing flexibility and productivity in Industry 4.0 production networks with autonomous mobile robots and smart intralogistics. *Annals of Operations Research*, *308*(1-2), 125–143. https://doi.org/10.1007/s10479-020-03526-7 (ver pp. 1, 7, 10).

Framinan, J. M., Perez-Gonzalez, P., Fernandez-Viagas, V., & Gonzalez, V. (2022). Assessing the potential of decentralised scheduling: An experimental study for the job shop case. *IFAC-PapersOnLine*, *55*(10), 2617–2622. https://doi.org/10.1016/j.ifacol.2022.10.104 (ver pp. 12, 19, 26, 28).

Gao, J., Zheng, X., Gao, F., Tong, X., & Han, Q. (2022). Heterogeneous Multitype Fleet Green Vehicle Path Planning of Automated Guided Vehicle with Time Windows in Flexible Manufacturing System. *Machines*, *10*(3), 197. https://doi.org/10.3390/machines10030197 (ver pp. 7, 10, 15, 16).

Ghaleb, M., & Taghipour, S. (2023). Dynamic shop-floor scheduling using real-time information: A case study from the thermoplastic industry. *Computers & Operations Research*, *152*, 106134. https://doi.org/10.1016/j.cor.2022.106134 (ver pp. 11, 22–25, 29, 31, 61, 92, 93, 187, 193).

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. Em *Annals of Discrete Mathematics* (pp. 287–326, Vol. 5). Elsevier. https://doi.org/10.1016/S0167-5060(08)70356-X (ver pp. 23, 55, 93).

Gurobi Optimization. (2023a). Gurobi Optimizer Reference Manual. https://www.gurobi.com/wp-content/plugins/hd_documentations/documentation/current/refman.pdf (ver pp. 2, 71–73, 94, 179–181, 187, 188).

Gurobi Optimization. (2023b). Our Story. Obtido 2023-09-29, de https://www.gurobi.com/company/our-story/ (ver p. 71).

Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2 (ver p. 41).

Hasan, H. A., Guéret, C., Lemoine, D., & Rivreau, D. (2022). A Rolling-Horizon Approach for a Surgery Case Scheduling Problem with Sterilizing Constraints. *IFAC-PapersOnLine*, *55*(10), 1625–1630. https://doi.org/10.1016/j.ifacol.2022.09.623 (ver pp. 12, 21, 23, 26, 29).

Hillier, F. S., & Lieberman, G. J. (2015). *Introduction to operations research* (Tenth edition). McGraw-Hill. (Ver pp. 1, 24, 60, 61, 71, 187).

Himmiche, S., Aubry, A., Marangé, P., & Pétin, J.-F. (2021). A framework for robust scheduling under stochastic perturbations. *IFAC-PapersOnLine*, *54*(1), 1168–1173. https://doi.org/10.1016/j.ifacol.2021.08.137 (ver pp. 12, 19, 22–24, 26–28).

Imeguisa Portugal. (2021). *Projeto AGiLE - Resiliência, otimização e eficiência no abastecimento em tempo-real de linhas de montagem por uma frota de robôs autónomos móveis* (Anexo Tecnico). Imeguisa Portugal. (Ver pp. 2, 34).

Imeguisa Portugal. (2022, julho). *Entregável 1.1 Documento com especificação detalhada de piloto a instalar na AE* (Entregável de Projecto). Imeguisa Portugal. (Ver pp. 35, 37).

Kelly, J. D., & Zyngier, D. (2007). An Improved MILP Modeling of Sequence-Dependent Switchovers for Discrete-Time Scheduling Problems. *Industrial & Engineering Chemistry Research*, *46*(14), 4964–4973. https://doi.org/10.1021/ie061572g (ver p. 92).

Konishi, S., & Kitagawa, G. (2008). *Information criteria and statistical modeling*. Springer. (Ver pp. 42, 43).

Kopp, T., Baumgartner, M., Seeger, M., & Kinkel, S. (2023). Perspectives of managers and workers on the implementation of automated-guided vehicles (AGVs)—a quantitative survey. *The International Journal of Advanced Manufacturing Technology*, *126*(11-12), 5259–5275. https://doi.org/10.1007/s00170-023-11294-4 (ver pp. 10, 17).

Kvam, P. H., & Vidakovic, B. (2007). *Nonparametric statistics with applications to science and engineering*. Wiley-Interscience.

Law, A. M. (2013). *Simulation Modeling and Analysis* (Fifth edition). McGraw-Hill Education. (Ver pp. 18, 39, 51, 76, 93).

Levorato, M., Figueiredo, R., & Frota, Y. (2022). Exact solutions for the two-machine robust flow shop with budgeted uncertainty. *European Journal of Operational Research*, *300*(1), 46–57. https://doi.org/10.1016/j.ejor.2021.10.021 (ver pp. 11, 20, 23–25, 31).

Levorato, M., Sotelo, D., Figueiredo, R., & Frota, Y. (2023). Robust permutation flow shop total weighted completion time problem: Solution and application to the oil and gas industry. *Computers & Operations Research*, *151*, 106117. https://doi.org/10.1016/j.cor.2022.106117 (ver pp. 1, 11, 20, 23–25, 28, 29).

Li, L., Li, Y., Liu, R., Zhou, Y., & Pan, E. (2023). A Two-stage Stochastic Programming for AGV scheduling with random tasks and battery swapping in automated container terminals. *Transportation Research Part E: Logistics and Transportation Review*, *174*, 103110. https://doi.org/10.1016/j.tre.2023.103110 (ver pp. 11, 25).

Li, Y.-Z., Pan, Q.-K., Gao, K.-Z., Tasgetiren, M. F., Zhang, B., & Li, J.-Q. (2021). A green scheduling algorithm for the distributed flowshop problem. *Applied Soft Computing*, *109*, 107526. https://doi.org/10.1016/j.asoc.2021.107526 (ver pp. 12, 19–21, 24).

Liu, J., & MacCarthy, B. (1997). A global MILP model for FMS scheduling. *European Journal of Operational Research*, *100*(3), 441–453. https://doi.org/10.1016/S0377-2217(96)00055-0 (ver p. 66).

Liu, L., Qu, T., Thürer, M., Ma, L., Zhang, Z., & Yuan, M. (2022). A new knowledge-guided multi-objective optimisation for the multi-AGV dispatching problem in dynamic production environments. *International Journal of Production Research*, *61*(17), 6030–6051. https://doi.org/10.1080/00207543.2022.2122619 (ver pp. 8, 10, 15, 17).

Lourenço, J. M. (2021). *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (ver p. iii).

Mahmood, K., Karjust, K., & Raamets, T. (2021). Production Intralogistics Automation Based on 3D Simulation Analysis. *Journal of Machine Engineering*, 101–115. https://doi.org/10.36897/jme/137081 (ver pp. 1, 7, 11, 15, 17, 18).

Manzini, M., Demeulemeester, E., & Urgo, M. (2022). A predictive–reactive approach for the sequencing of assembly operations in an automated assembly line. *Robotics and Computer-Integrated Manufacturing*, *73*, 102201. https://doi.org/10.1016/j.rcim.2021.102201 (ver pp. 12, 20, 22, 24, 25, 27, 29).

McKinney, W. (2010). Data structures for statistical computing in python. *Proceedings of the 9th Python in Science Conference*, *445*, 51–56 (ver p. 41).

Meng, L., Gao, K., Ren, Y., Zhang, B., Sang, H., & Chaoyong, Z. (2022). Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, *71*, 101058. https://doi.org/10.1016/j.swevo.2022.101058 (ver p. 92).

Meng, L., Zhang, B., Gao, K., & Duan, P. (2022). An MILP Model for Energy-Conscious Flexible Job Shop Problem with Transportation and Sequence-Dependent Setup Times. *Sustainability*, *15*(1), 776. https://doi.org/10.3390/su15010776 (ver p. 92).

Noh, H. Y., & Fagert, J. (2022, janeiro). Big data analysis for civil infrastructure sensing. Em J. P. Lynch, H. Sohn & M. L. Wang (Eds.), *Sensor Technologies for Civil Infrastructures (Second Edition)* (pp. 639–677). Woodhead Publishing. https://doi.org/10.1016/B978-0-08-102706-6.00007-6

Novak, A., Sucha, P., Novotny, M., Stec, R., & Hanzalek, Z. (2022). Scheduling jobs with normally distributed processing times on parallel machines. *European Journal of Operational Research*, *297*(2), 422–441. https://doi.org/10.1016/j.ejor.2021.05.011 (ver pp. 11, 20, 21, 29, 31).

Ohno, T., & Bodek, N. (2008). *Toyota production system: beyond large-scale production* (Reprinted). Productivity Press. (Ver p. 33).

Oyekanlu, E. A., Smith, A. C., Thomas, W. P., Mulroy, G., Hitesh, D., Ramsey, M., Kuhn, D. J., Mcghinnis, J. D., Buonavita, S. C., Looper, N. A., Ng, M., Ng'oma, A., Liu, W., Mcbride, P. G., Shultz, M. G., Cerasi, C., & Sun, D. (2020). A Review of Recent Advances in Automated Guided Vehicle Technologies: Integration Challenges and Research Areas for 5G-Based Smart Manufacturing Applications. *IEEE Access*, *8*, 202312–202353. https://doi.org/10.1109/ACCESS.2020.3035729 (ver pp. 9, 11, 15, 16).

Padberg, M., & Rinaldi, G. (1991). A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, *33*(1), 60–100. https://doi.org/10.1137/1033004 (ver p. 72).

Panzer, M., Bender, B., & Gronau, N. (2022). Neural agent-based production planning and control: An architectural review. *Journal of Manufacturing Systems*, *65*, 743–766. https://doi.org/10.1016/j.jmsy.2022.10.019 (ver pp. 11, 20).

Pardalos, P. M., Du, D.-Z., & Graham, R. L. (Eds.). (2013). *Handbook of Combinatorial Optimization*. Springer New York. https://doi.org/10.1007/978-1-4419-7997-1 (ver p. 186).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (2011). Scikit-learn: Machine Learning in Python. *Machine Learning In Python*, *12*, 2825–2830.

Peng, W., Lin, X., & Li, H. (2023). Critical chain based Proactive-Reactive scheduling for Resource-Constrained project scheduling under uncertainty. *Expert Systems with Applications*, *214*, 119188. https://doi.org/10.1016/j.eswa.2022.119188 (ver pp. 11, 19).

Pinedo, M. (2007). *Planning and scheduling in manufacturing and services* (Corr. as of the 3. print). Springer. (Ver p. 54).

Pinedo, M. (2016). *Scheduling: Theory, Algorithms, and Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3 (ver pp. 20, 21, 23, 65).

Pinto, G., Silva, F., Campilho, R., Casais, R., Fernandes, A., & Baptista, A. (2019). Continuous improvement in maintenance: a case study in the automotive industry involving Lean tools. *Procedia Manufacturing*, *38*, 1582–1591. https://doi.org/10.1016/j.promfg.2020.01.127 (ver pp. 18, 31).

Python Software Foundation & JetBrains. (2022). Python Developers Survey 2022 Results. Obtido 2023-09-29, de https://lp.jetbrains.com/python-developers-survey-2022/ (ver p. 72).

Razali, N. M., & Wah, Y. B. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, *2*(1), 21–33 (ver p. 42).

Reis, W. P. N. D., Couto, G. E., & Junior, O. M. (2023). Automated guided vehicles position control: a systematic literature review. *Journal of Intelligent Manufacturing*, *34*(4), 1483–1545. https://doi.org/10.1007/s10845-021-01893-x (ver pp. 10, 15).

Reith, K.-B., Rank, S., & Schmidt, T. (2021). Conflict-minimal routing for free-ranging transportation vehicles in in-house logistics based on an a-priori lane design. *Journal of Manufacturing Systems*, *61*, 97–111. https://doi.org/10.1016/j.jmsy.2021.07.019 (ver pp. 10, 15, 16).

Schlecht, M., Himmiche, S., Goepp, V., De Guio, R., & Köbler, J. (2022). Data-driven decision process for robust scheduling of remanufacturing systems. *IFAC-PapersOnLine*, *55*(10), 755–760. https://doi.org/10.1016/j.ifacol.2022.09.500 (ver pp. 11, 19, 27).

Sierra-García, J. E., Fernández-Rodríguez, V., Santos, M., & Quevedo, E. (2023). Development and Experimental Validation of Control Algorithm for Person-Following Autonomous Robots. *Electronics*, *12*(9), 2077. https://doi.org/10.3390/electronics12092077 (ver pp. 10, 17).

Sim, C. H., Gan, F. F., & Chang, T. C. (2005). Outlier Labeling With Boxplot Procedures. *Journal of the American Statistical Association*, *100*(470), 642–652. https://doi.org/10.1198/016214504000001466 (ver p. 39).

Simon, H. A. (1969). *The Sciences of the Artificial* (3. ed., [Nachdr.]). MIT Press. (Ver p. 4).

Software, F. (2023). Welcome to FlexSim. Obtido 2023-08-21, de https://docs.flexsim.com/en/23.2/Introduction/Welcome/Welcome.html (ver p. 28).

Souza, R. L. C., Ghasemi, A., Saif, A., & Gharaei, A. (2022). Robust job-shop scheduling under deterministic and stochastic unavailability constraints due to preventive and corrective maintenance. *Computers & Industrial Engineering*, *168*, 108130. https://doi.org/10.1016/j.cie.2022.108130 (ver pp. 12, 20, 21, 24–26, 28, 29, 31).

Sperling, M., Schulz, B., Enke, C., Giebels, D., & Furmans, K. (2023). Classified AGV Material Flow and Layout Data Set for Multidisciplinary Investigation. *IEEE Access*, *11*, 94992–95007. https://doi.org/10.1109/ACCESS.2023.3308216 (ver pp. 10, 16).

Srikar, B. N., & Ghosh, S. (1986). A MILP model for the *n* -job, *M* -stage flowshop with sequence dependent set-up times. *International Journal of Production Research*, *24*(6), 1459–1474. https://doi.org/10.1080/00207548608919815 (ver p. 92).

Stączek, P., Pizoń, J., Danilczuk, W., & Gola, A. (2021). A Digital Twin Approach for the Improvement of an Autonomous Mobile Robots (AMR's) Operating Environment—A Case Study. *Sensors*, *21*(23), 7830. https://doi.org/10.3390/s21237830 (ver pp. 7, 10, 15–19).

Sui, Y., & Wang, Z. (2022). Parallel Identical Machines Scheduling to Minimize the Maximum Inter-completion Time with Uncertain Processing Time. *IFAC-PapersOnLine*, *55*(10), 2599–2604. https://doi.org/10.1016/j.ifacol.2022.10.101 (ver pp. 11, 20, 21).

Torres-Carrion, P. V., Gonzalez-Gonzalez, C. S., Aciar, S., & Rodriguez-Morales, G. (2018). Methodology for systematic literature review applied to engineering and education. *2018 IEEE Global Engineering Education Conference (EDUCON)*, 1364–1373. https://doi.org/10.1109/EDUCON.2018.8363388 (ver p. 8).

Tukey, J. W. (1977). *Exploratory data analysis*. Addison-Wesley Pub. Co. (Ver p. 39).

Van Eck, N. J., & Waltman, L. (2010). Software survey: VOSviewer, a computer program for bibliometric mapping. *Scientometrics*, *84*(2), 523–538. https://doi.org/10.1007/s11192-009-0146-3 (ver p. 13).

van Eck, N. J., & Waltman, L. (2022, janeiro). VOSviewer Manual. https://www.vosviewer.com/documentation/Manual_VOSviewer_1.6.8.pdf (ver p. 13).

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Van Der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . Vázquez-Baeza, Y. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2 (ver p. 42).

Vom Brocke, J., Hevner, A., & Maedche, A. (2020). Introduction to Design Science Research [Series Title: Progress in IS]. Em J. Vom Brocke, A. Hevner & A. Maedche (Eds.), *Design Science Research. Cases* (pp. 1–13). Springer International Publishing. https://doi.org/10.1007/978-3-030-46781-4_1 (ver p. 3).

Wan, H., Wang, H., Scotney, B., & Liu, J. (2019). A Novel Gaussian Mixture Model for Classification. *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 3298–3303. https://doi.org/10.1109/SMC.2019.8914215 (ver p. 41).

Waskom, M. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021 (ver p. 41).

Wilson, R. J. (2009). *Introduction to graph theory* (4. ed., [Nachdr.]). Prentice Hall. (Ver p. 65).

Yalaoui, F., & Quy Nguyen, N. (2021). Identical Machine Scheduling Problem with Sequence-Dependent Setup Times: MILP Formulations Computational Study. *American Journal of Operations Research*, *11*(01), 15–34. https://doi.org/10.4236/ajor.2021.111002 (ver p. 92).

Zimmermann, E., Mezgebe, T. T., Bril El Haouzi, H., Thomas, P., Pannequin, R., & Noyel, M. (2021). Multicriteria decision-making method for scheduling problem based on smart batches and their quality prediction capability. *Computers in Industry*, *133*, 103549. https://doi.org/10.1016/j.compind.2021.103549 (ver pp. 12, 20).

Zuur, A. F., Ieno, E. N., Walker, N., Saveliev, A. A., & Smith, G. M. (2009). *Mixed effects models and extensions in ecology with R*. Springer New York. https://doi.org/10.1007/978-0-387-87458-6 (ver p. 62).

APÊNDICE A

MODELAÇÃO PROPOSTA DO *LAYOUT* DA LINHA DE MONTAGEM

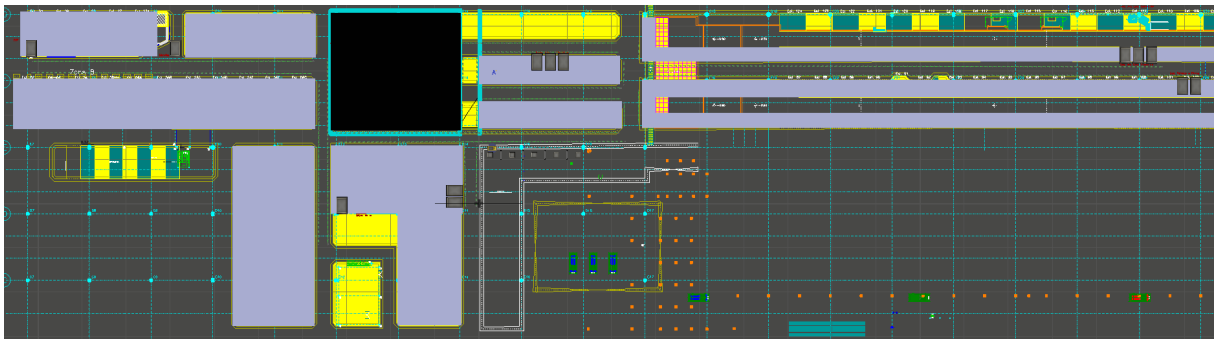


Figura A.1: Modelação atual do *Layout* da linha de montagem final da Volkswagen Autoeuropa, Lda.

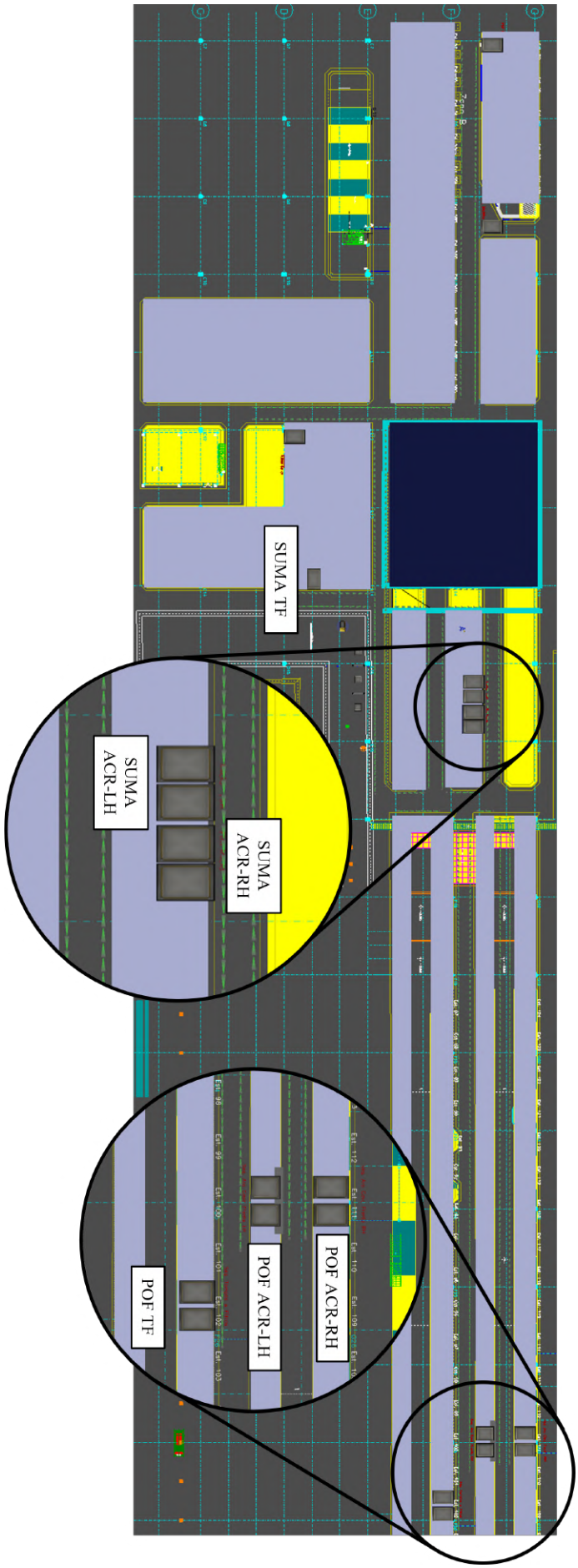


Figura A.2: Modelação proposta do *Layout* da linha de montagem final da Volkswagen Autoeuropa, Lda.

APÊNDICE B

ANÁLISE DOS TEMPOS DE *PICKING*

B.1 Gráficos *box-plot* para deteção de *outliers*

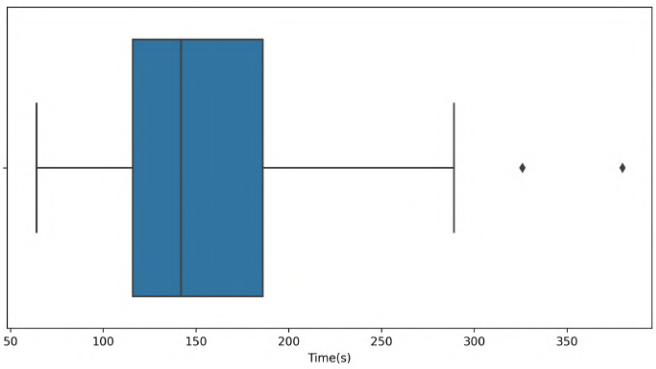


Figura B.1: Gráfico *box-plot*: ACR_LH_M1

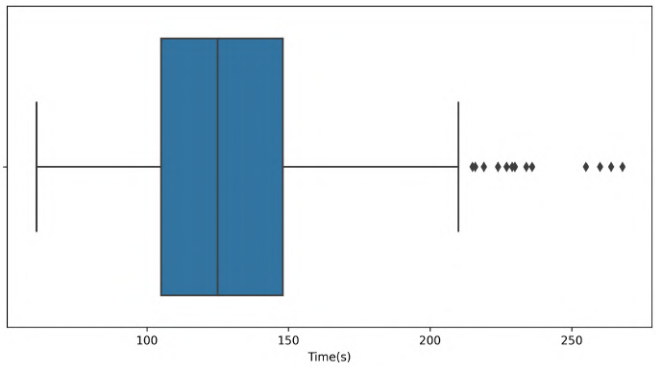


Figura B.2: Gráfico *box-plot*: ACR_LH_M2

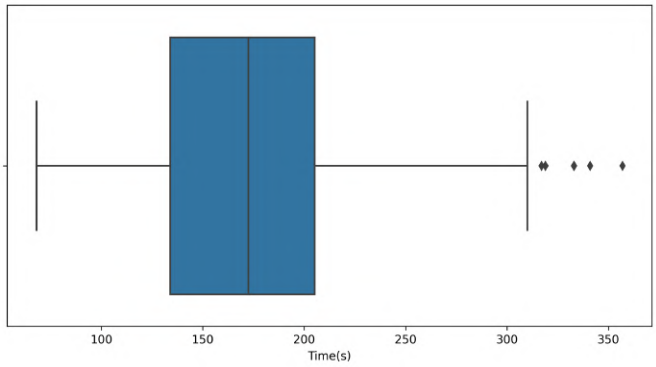


Figura B.3: Gráfico *box-plot*: ACR_LH_M3

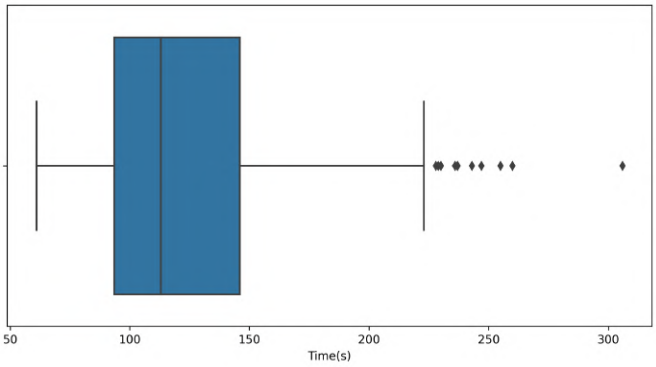


Figura B.4: Gráfico *box-plot*: ACR_LH_M4

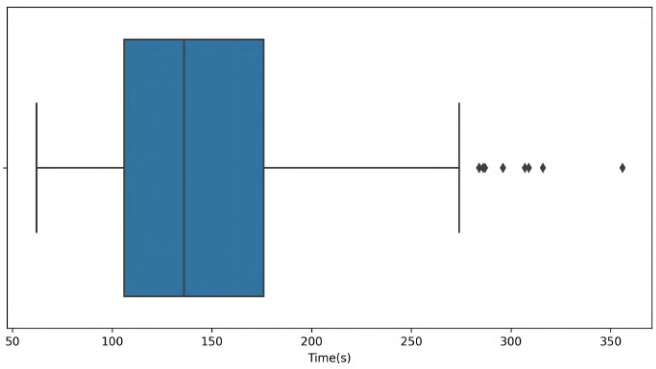


Figura B.5: Gráfico *box-plot*: ACR_RH_M1

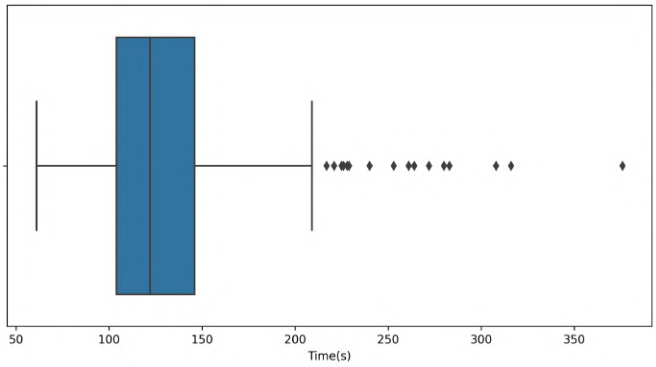


Figura B.6: Gráfico *box-plot*: ACR_RH_M2

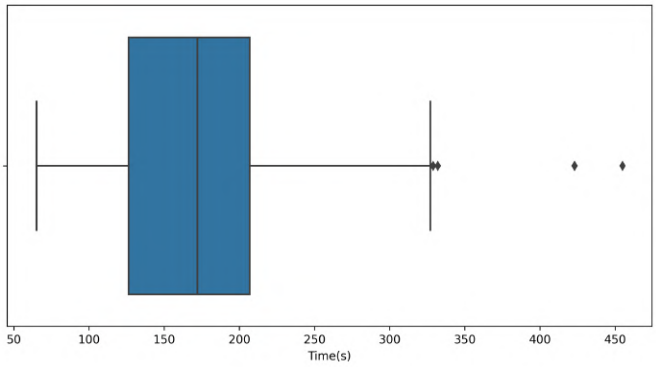


Figura B.7: Gráfico *box-plot*: ACR_RH_M3

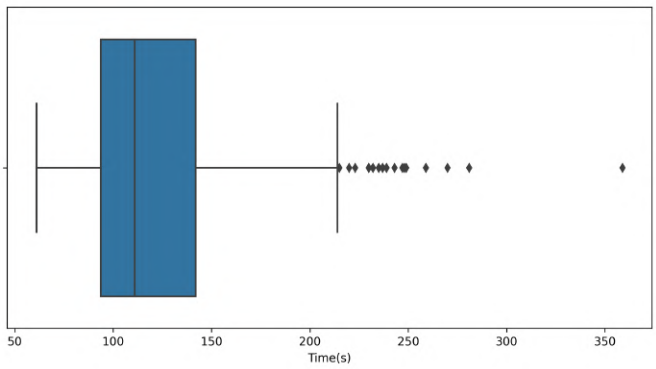


Figura B.8: Gráfico *box-plot*: ACR_RH_M4

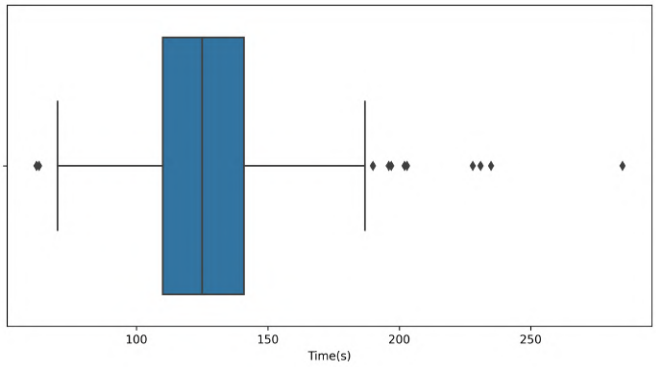


Figura B.9: Gráfico *box-plot*: MM_M1

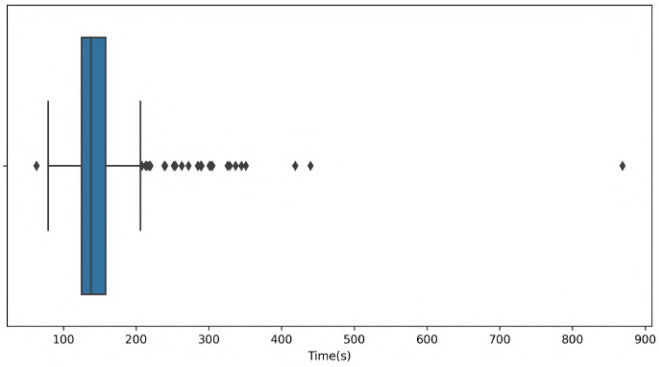


Figura B.10: Gráfico *box-plot*: MM_M2

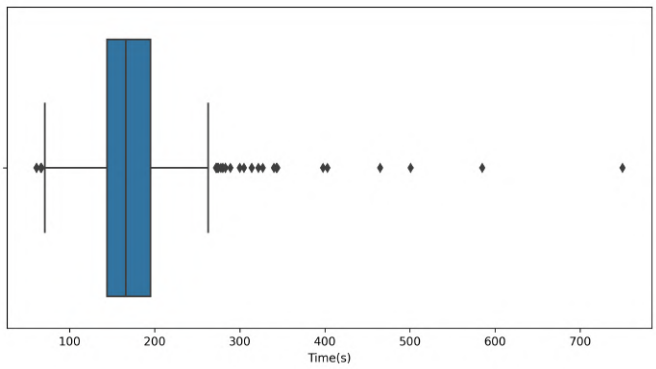


Figura B.11: Gráfico *box-plot*: MM_M3

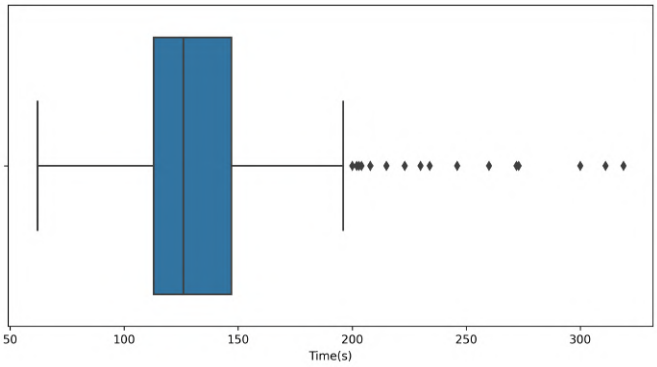


Figura B.12: Gráfico *box-plot*: MM_M4

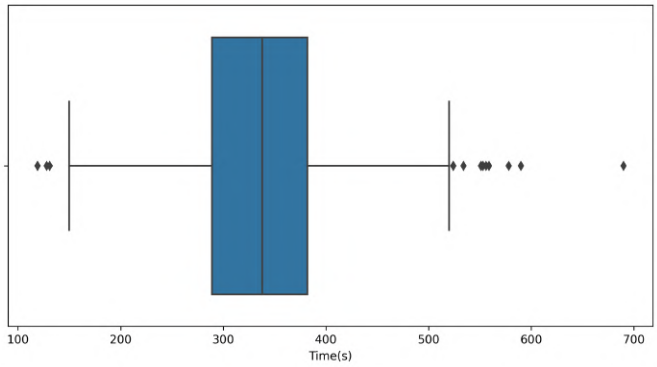


Figura B.13: Gráfico *box-plot*: TF_M1

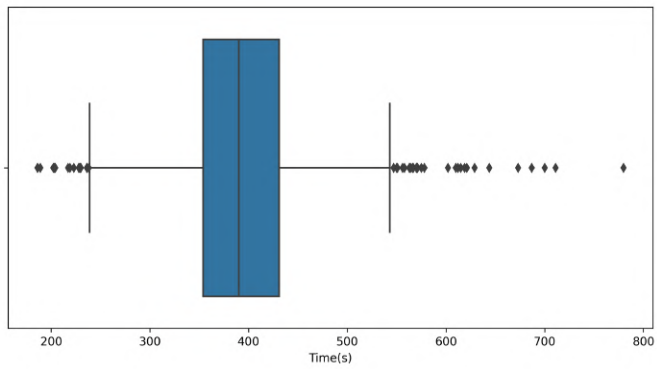


Figura B.14: Gráfico *box-plot*: TF_M2

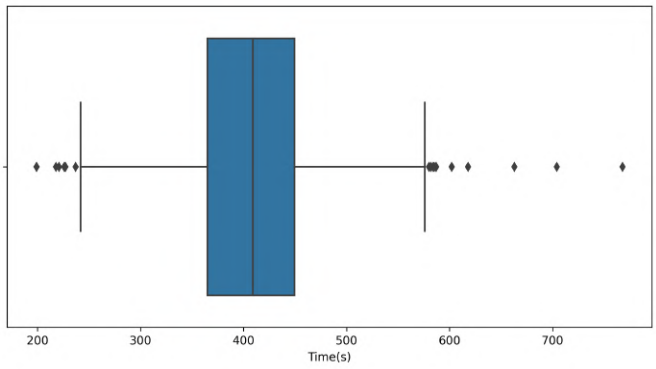


Figura B.15: Gráfico *box-plot*: TF_M3

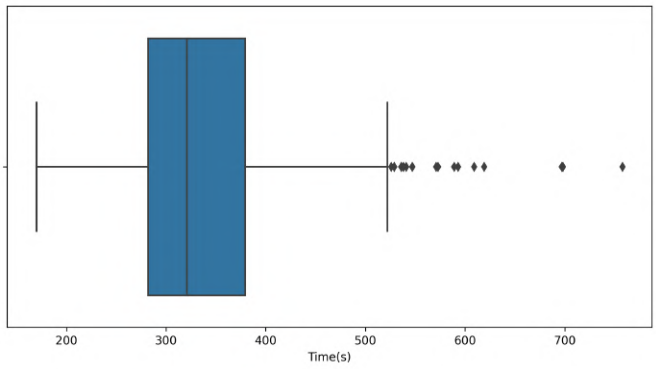


Figura B.16: Gráfico *box-plot*: TF_M4

Tabela B.1: Média e desvio padrão dos tempos de *picking*, após deteção e eliminação de *outliers*. A coluna n indica a dimensão da amostra inicial, antes da exclusão de tempos inferiores a 60 segundos e identificação de *outliers*. A coluna n_f indica a dimensão da amostra, após a exclusão de tempos inferiores a 60 segundos e identificação de *outliers*

Dados Originais (sem *outliers*)						
Turno	n	n_f	**Média (s)**	**Desvio padrão (s)**	**Mínimo (s)**	**Máximo (s)**
ACR_LH_M1	540	526	151,56	48,47	64	289
ACR_LH_M2	535	515	127,19	29,70	61	210
ACR_LH_M3	490	481	171,86	48,78	68	310
ACR_LH_M4	533	479	119,60	36,81	61	223
ACR_RH_M1	535	506	142,47	46,76	62	274
ACR_RH_M2	541	515	124,15	29,11	61	209
ACR_RH_M3	490	478	170,12	51,34	65	327
ACR_RH_M4	533	436	116,22	34,41	61	214
MM_M1	533	414	125,69	22,65	70	187
MM_M2	537	499	139,76	22,88	79	206
MM_M3	492	455	167,84	37,54	71	263
MM_M4	533	502	142,47	46,76	62	196
TF_M1	528	513	336,14	65,75	150	520
TF_M2	542	497	389,10	58,70	239	543
TF_M3	484	459	408,52	67,33	242	576
TF_M4	534	513	330,48	66,93	170	522

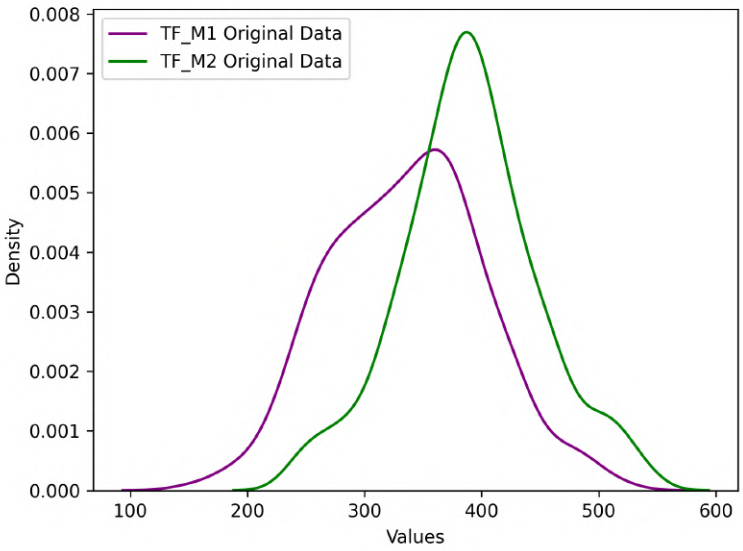


Figura B.17: Estimativas KDE dos dados originais sem *outliers* dos turnos TF_M1 e TF_M2

B.2 Histograma e estimativa KDE dos tempos de *picking*

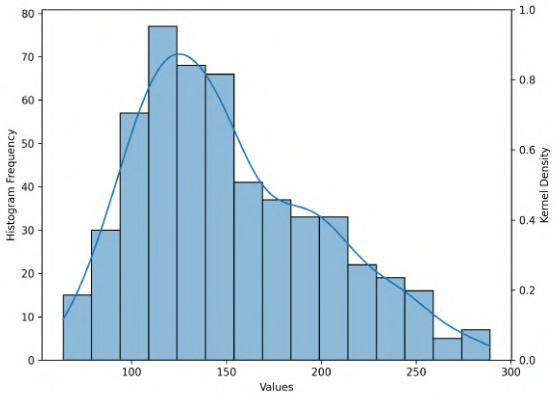


Figura B.18: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_LH_M1

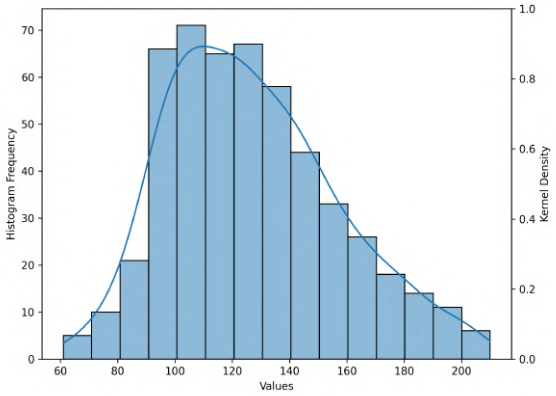


Figura B.19: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_LH_M2

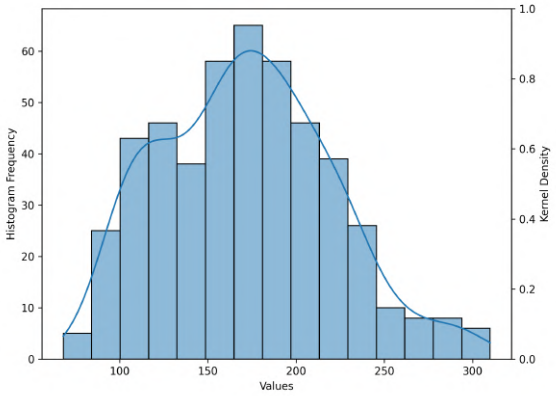


Figura B.20: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_LH_M3

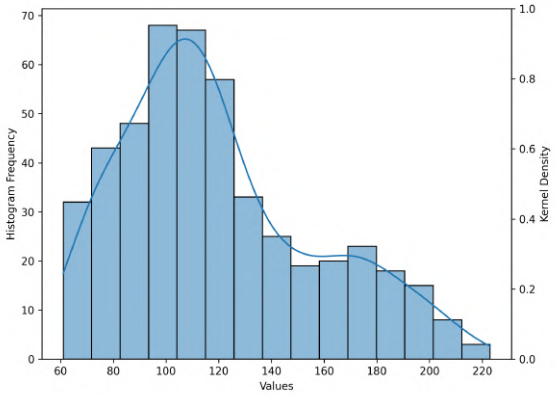


Figura B.21: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_LH_M4

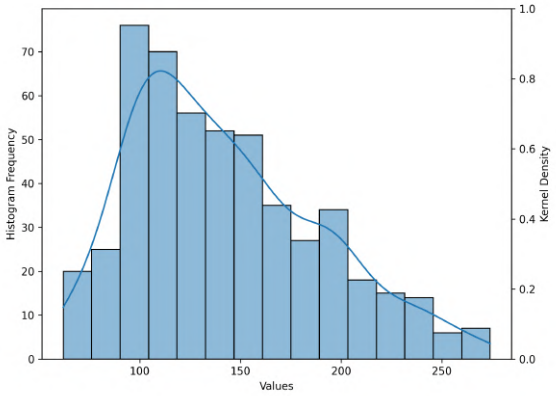


Figura B.22: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_RH_M1

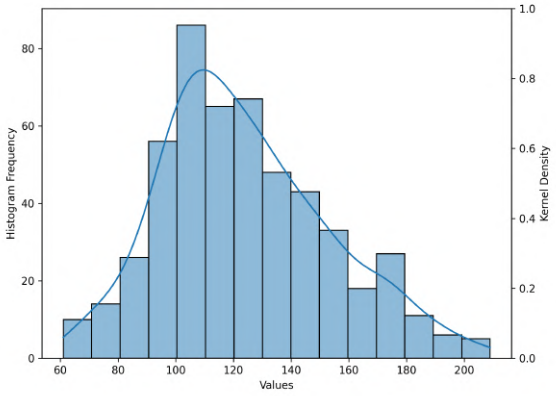


Figura B.23: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_RH_M2

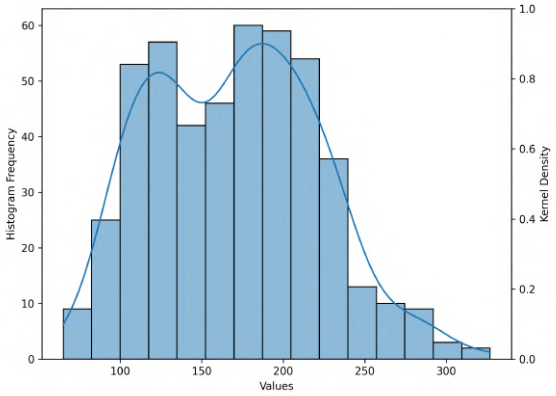


Figura B.24: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_RH_M3

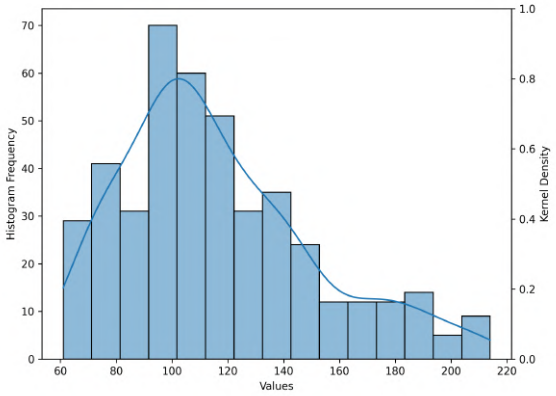


Figura B.25: Histograma e respetiva KDE dos dados originais sem *outliers*: ACR_RH_M4

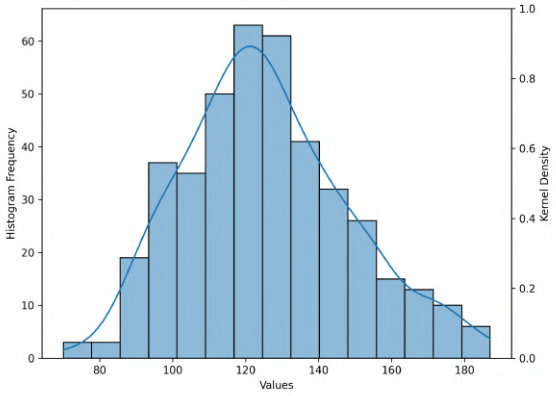


Figura B.26: Histograma e respetiva KDE dos dados originais sem *outliers*: MM_M1

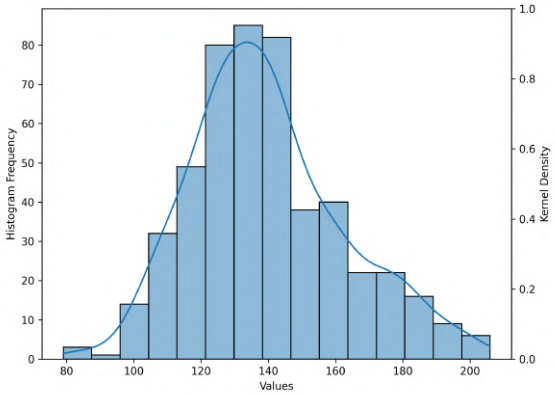


Figura B.27: Histograma e respetiva KDE dos dados originais sem *outliers*: MM_M2

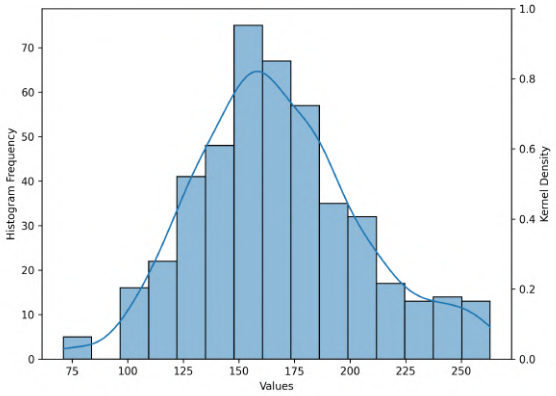


Figura B.28: Histograma e respetiva KDE dos dados originais sem *outliers*: MM_M3

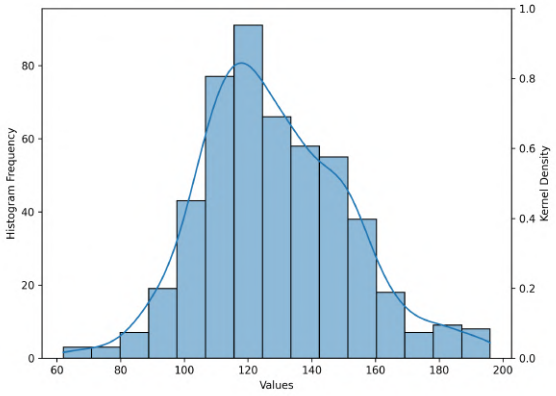


Figura B.29: Histograma e respetiva KDE dos dados originais sem *outliers*: MM_M4

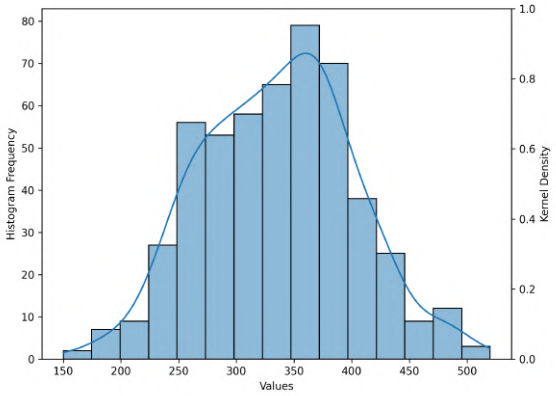


Figura B.30: Histograma e respetiva KDE dos dados originais sem *outliers*: TF_M1

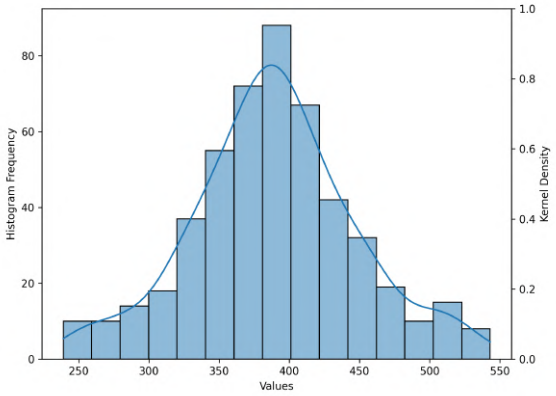


Figura B.31: Histograma e respetiva KDE dos dados originais sem *outliers*: TF_M2

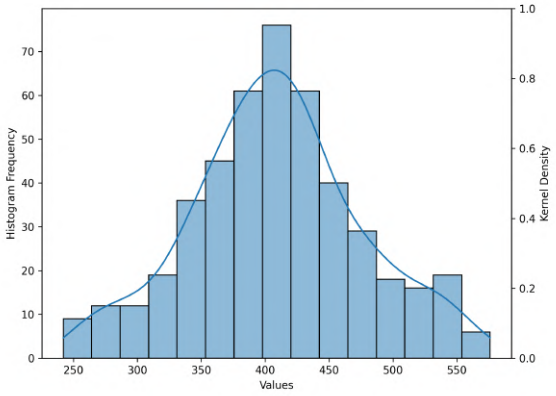


Figura B.32: Histograma e respetiva KDE dos dados originais sem *outliers*: TF_M3

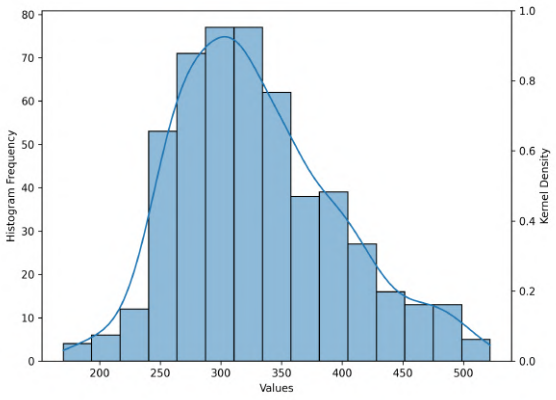
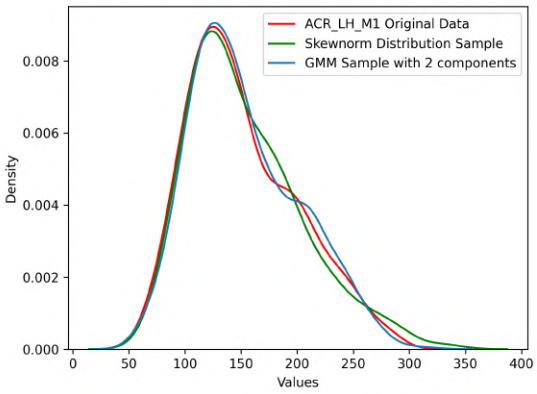
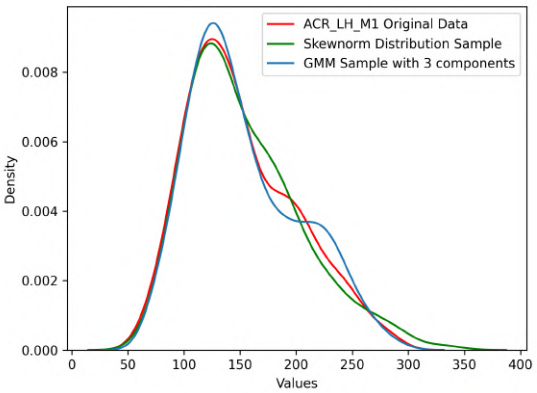


Figura B.33: Histograma e respetiva KDE dos dados originais sem *outliers*: TF_M4

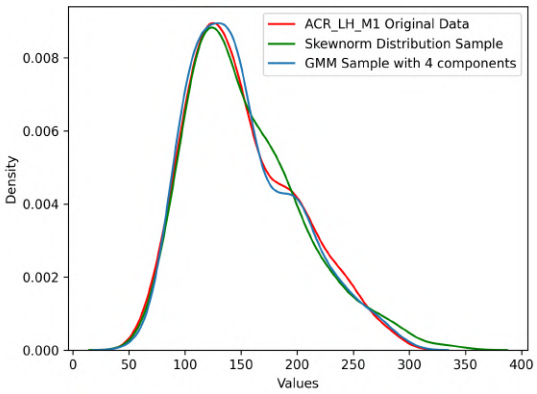
B.3 Estimativas KDE das amostras das distribuições ajustadas



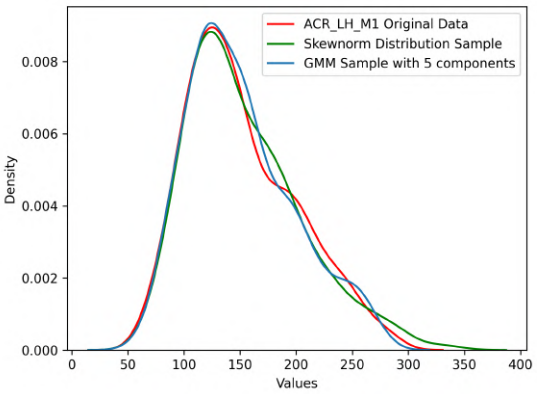
(a) *Gaussian Mixture Model* ou Modelo de Mistura Gaussiana (GMM) com 2 componentes



(b) GMM com 3 componentes

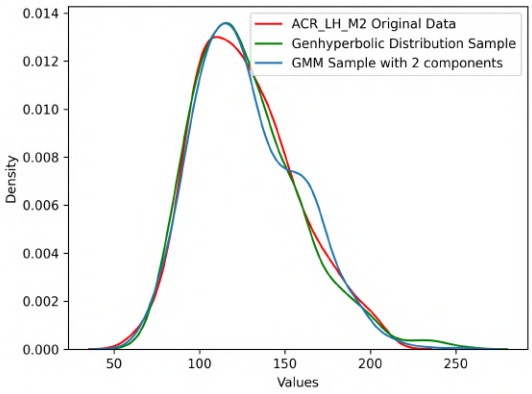


(c) GMM com 4 componentes

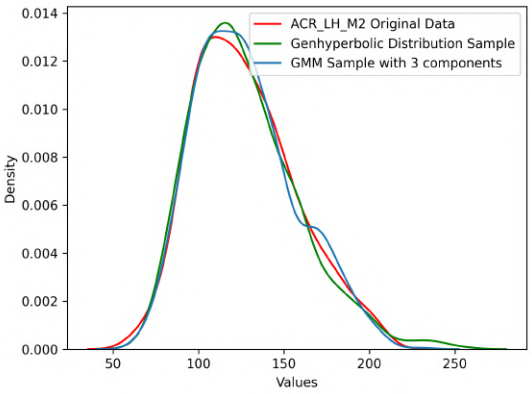


(d) GMM com 5 componentes

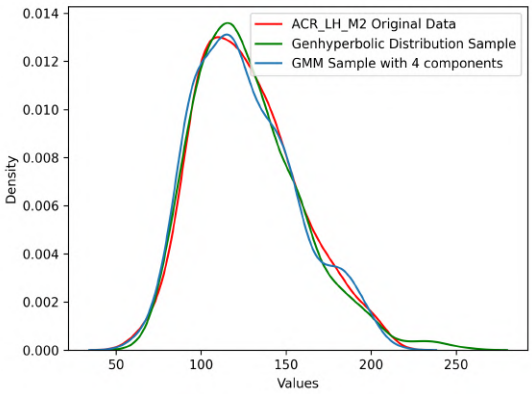
Figura B.34: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_LH_M1



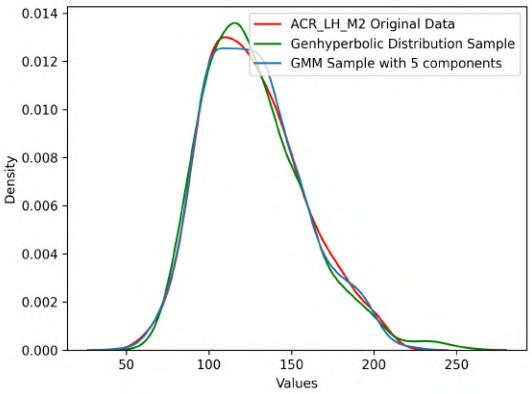
(a) GMM com 2 componentes



(b) GMM com 3 componentes

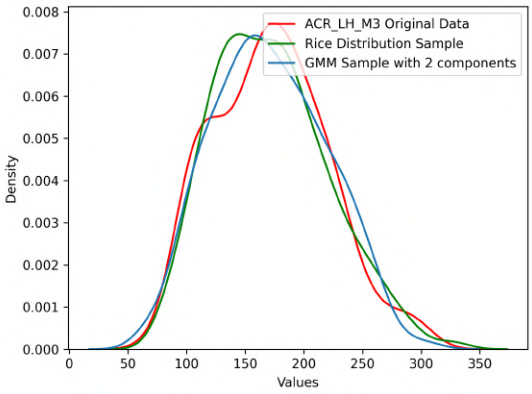


(c) GMM com 4 componentes

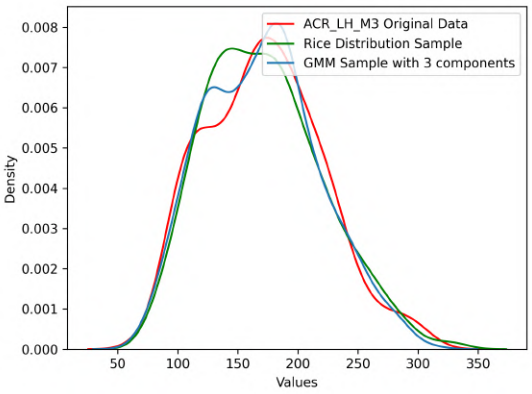


(d) GMM com 5 componentes

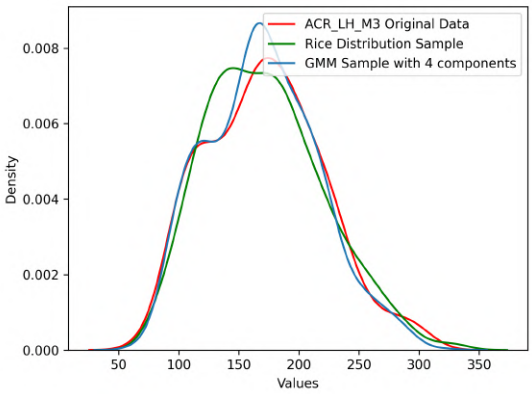
Figura B.35: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_LH_M2



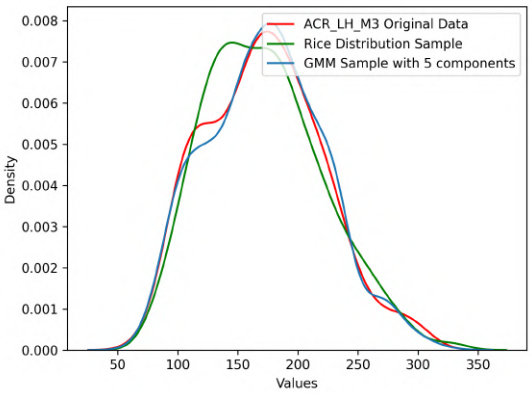
(a) GMM com 2 componentes



(b) GMM com 3 componentes

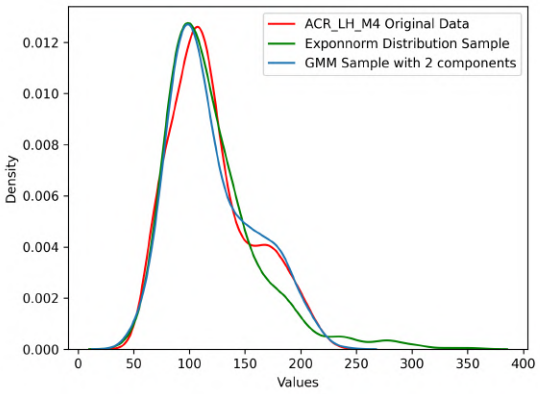


(c) GMM com 4 componentes

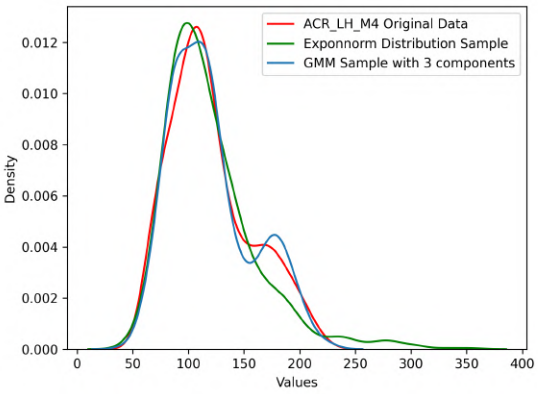


(d) GMM com 5 componentes

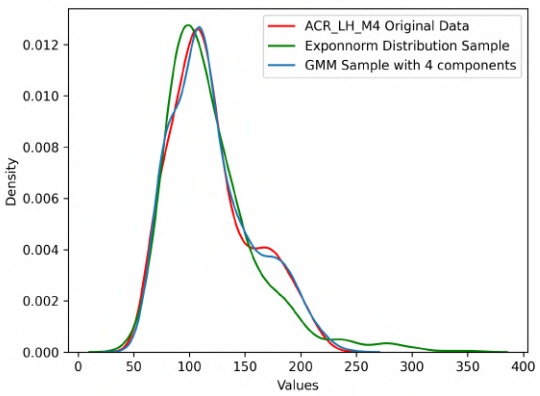
Figura B.36: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_LH_M3



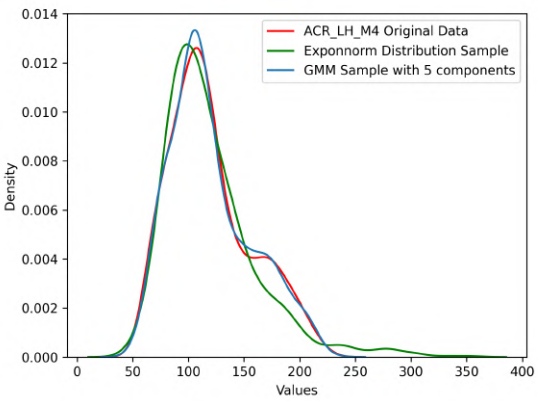
(a) GMM com 2 componentes



(b) GMM com 3 componentes

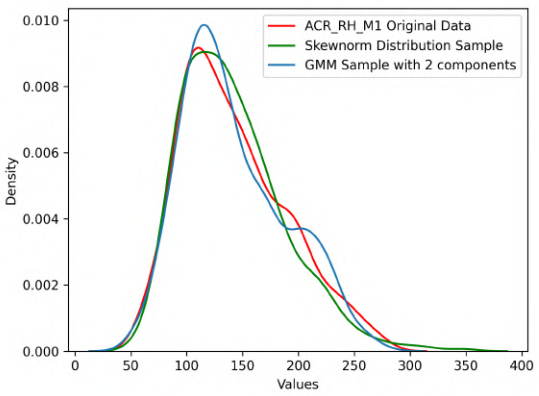


(c) GMM com 4 componentes

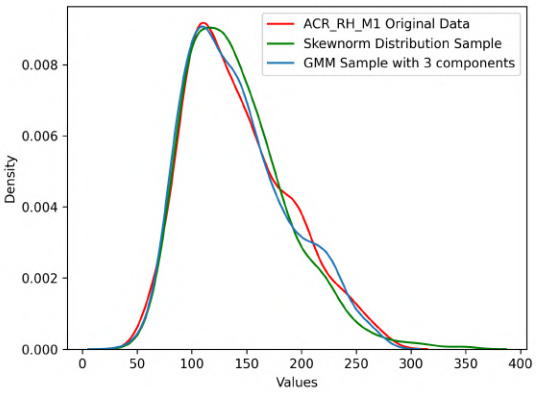


(d) GMM com 5 componentes

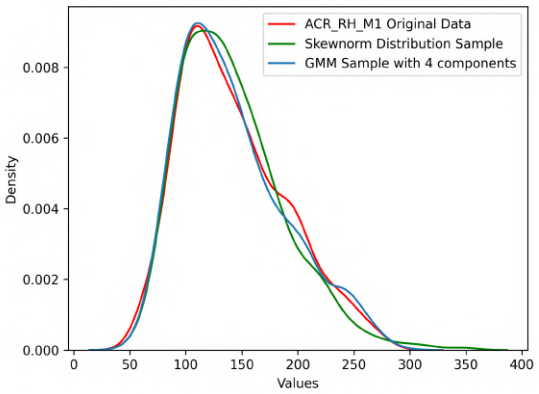
Figura B.37: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_LH_M4



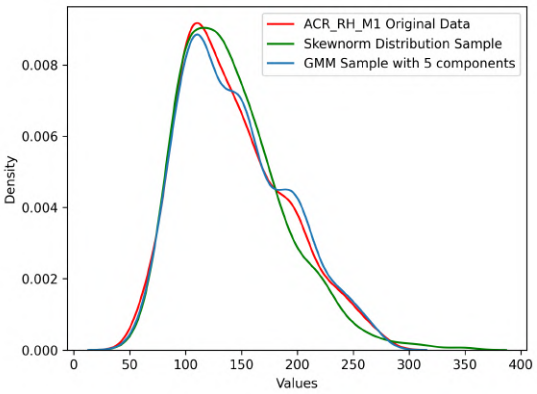
(a) GMM com 2 componentes



(b) GMM com 3 componentes

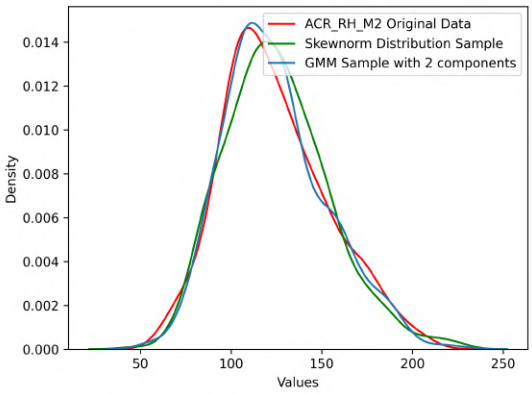


(c) GMM com 4 componentes

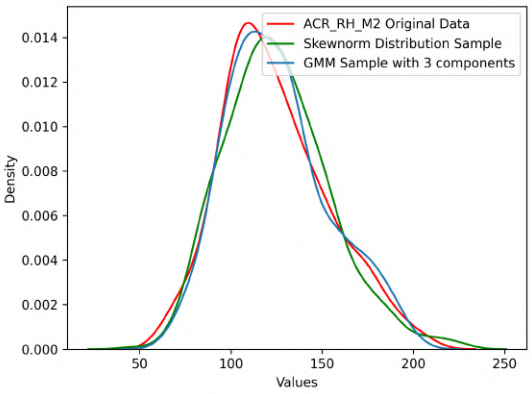


(d) GMM com 5 componentes

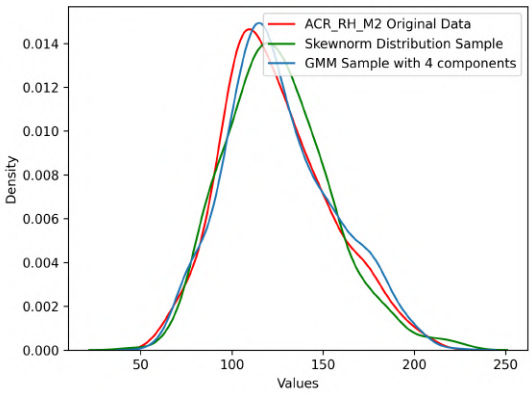
Figura B.38: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_RH_M1



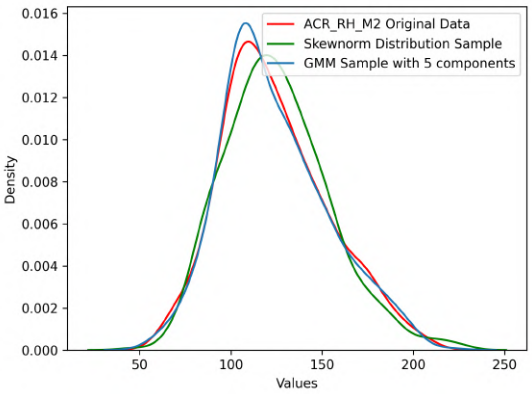
(a) GMM com 2 componentes



(b) GMM com 3 componentes

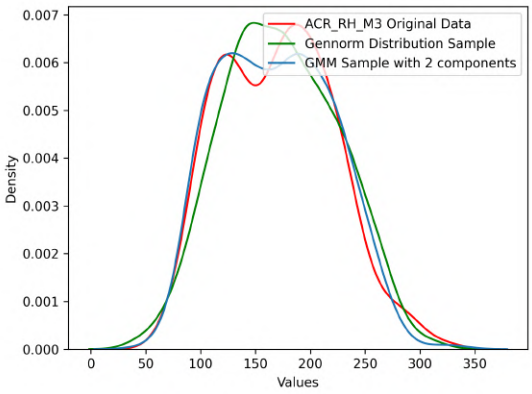


(c) GMM com 4 componentes

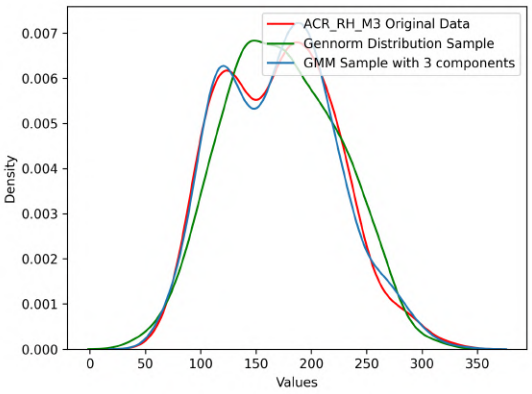


(d) GMM com 5 componentes

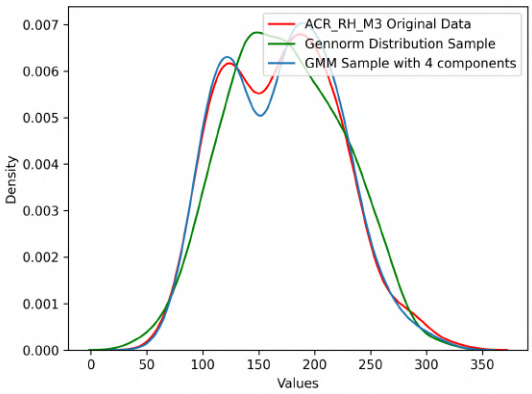
Figura B.39: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_RH_M2



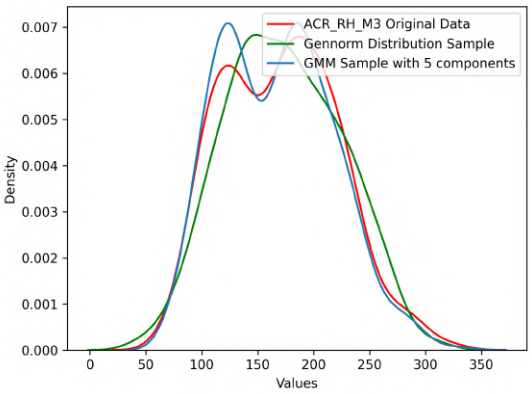
(a) GMM com 2 componentes



(b) GMM com 3 componentes

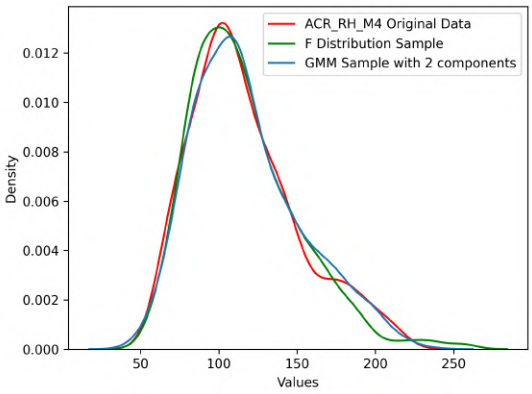


(c) GMM com 4 componentes

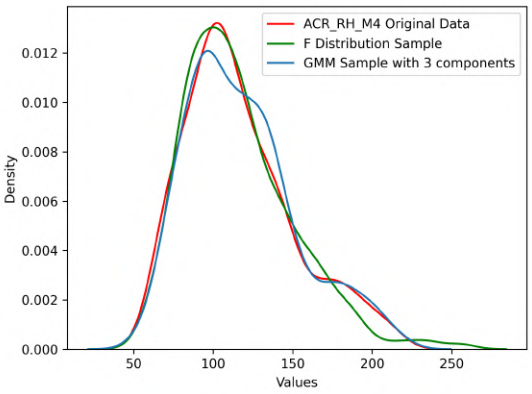


(d) GMM com 5 componentes

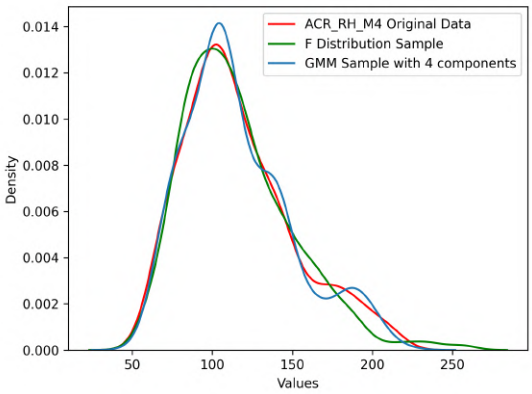
Figura B.40: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_RH_M3



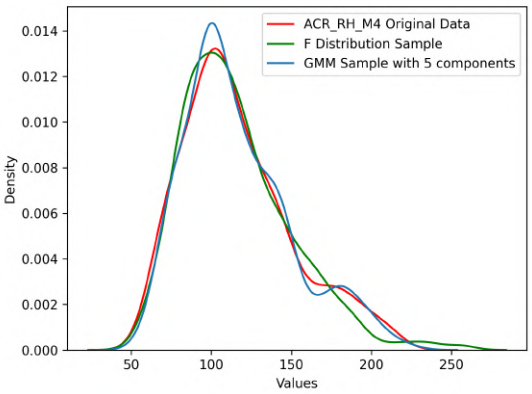
(a) GMM com 2 componentes



(b) GMM com 3 componentes

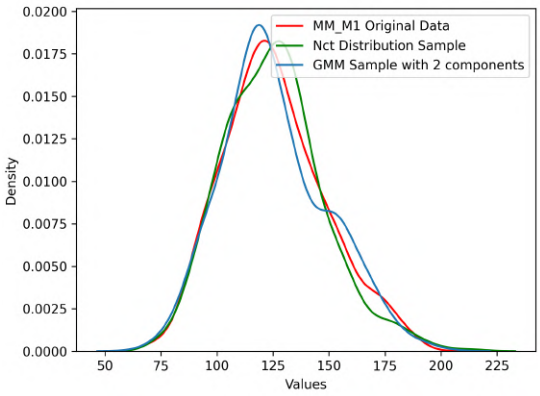


(c) GMM com 4 componentes

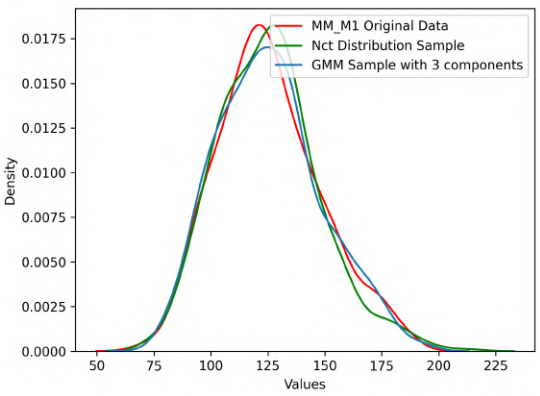


(d) GMM com 5 componentes

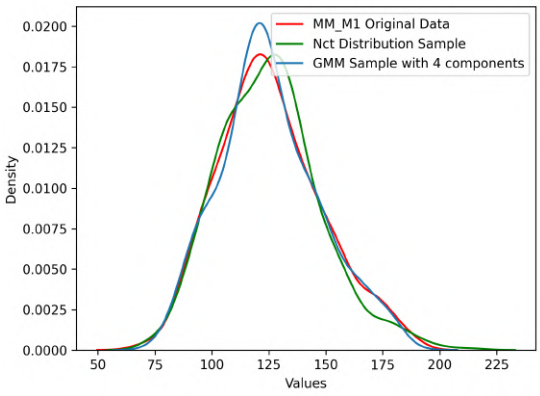
Figura B.41: Comparação das Estimativas KDE das distribuições ajustadas: turno ACR_RH_M4



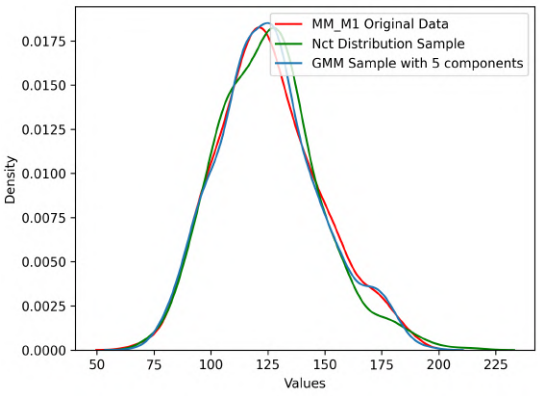
(a) GMM com 2 componentes



(b) GMM com 3 componentes

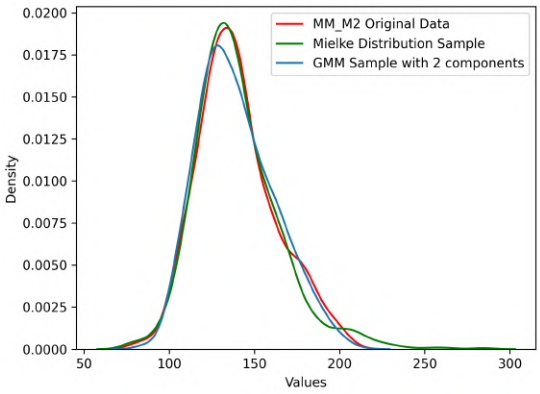


(c) GMM com 4 componentes

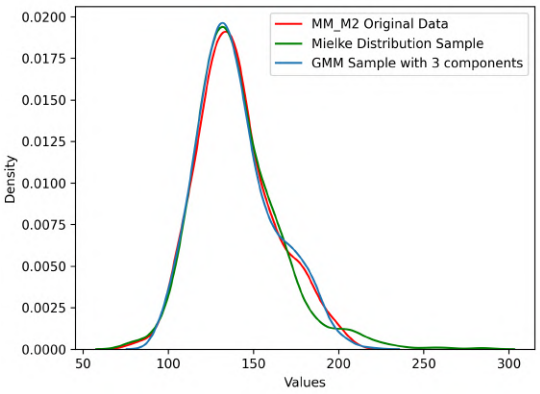


(d) GMM com 5 componentes

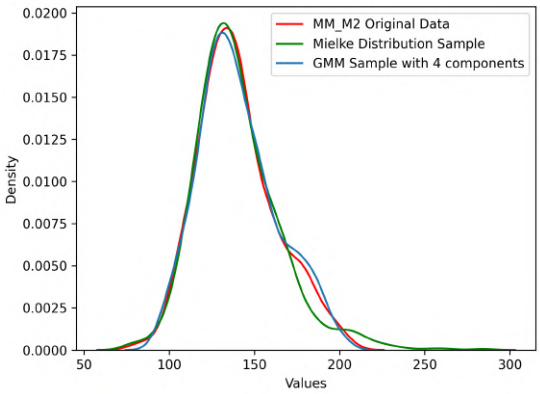
Figura B.42: Comparação das Estimativas KDE das distribuições ajustadas: turno MM_M1



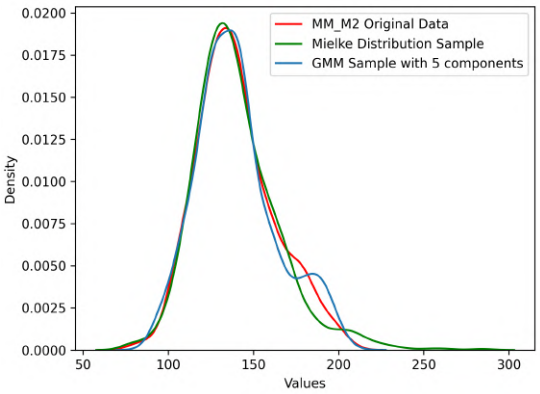
(a) GMM com 2 componentes



(b) GMM com 3 componentes

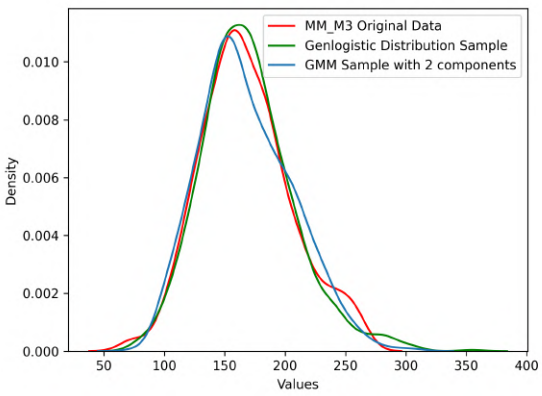


(c) GMM com 4 componentes

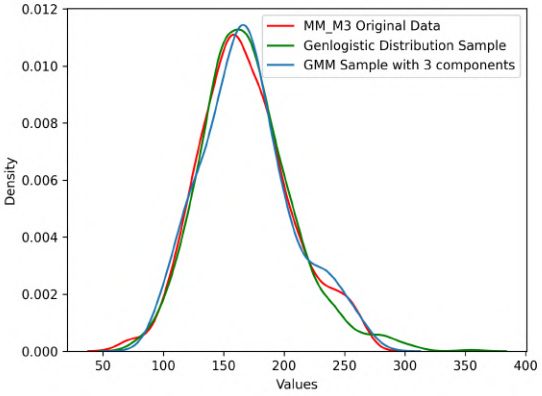


(d) GMM com 5 componentes

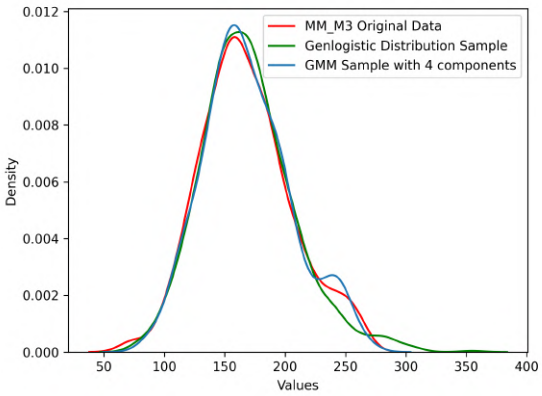
Figura B.43: Comparação das Estimativas KDE das distribuições ajustadas: turno MM_M2



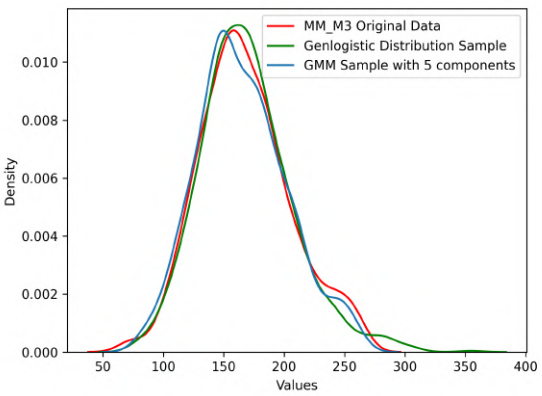
(a) GMM com 2 componentes



(b) GMM com 3 componentes

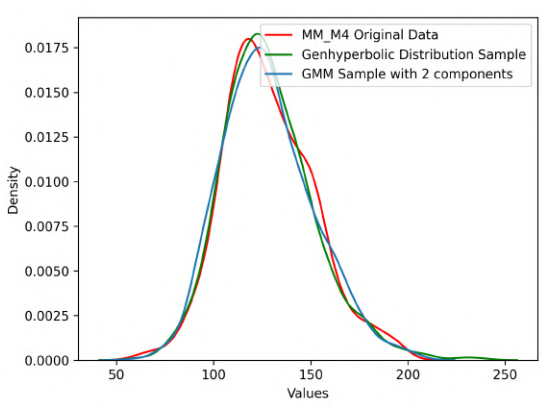


(c) GMM com 4 componentes

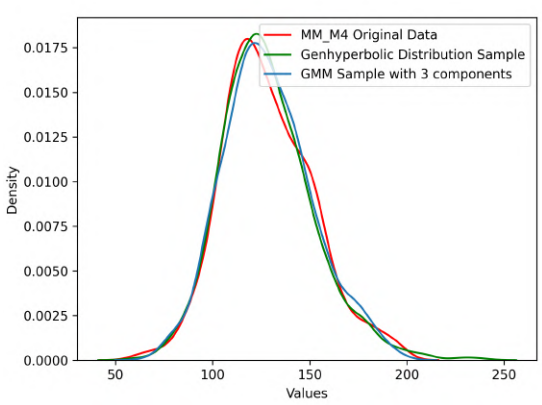


(d) GMM com 5 componentes

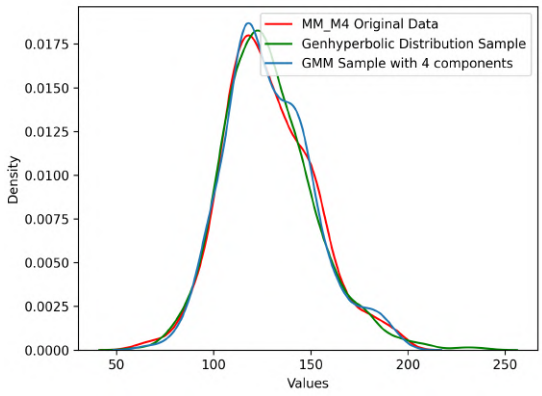
Figura B.44: Comparação das Estimativas KDE das distribuições ajustadas: turno MM_M3



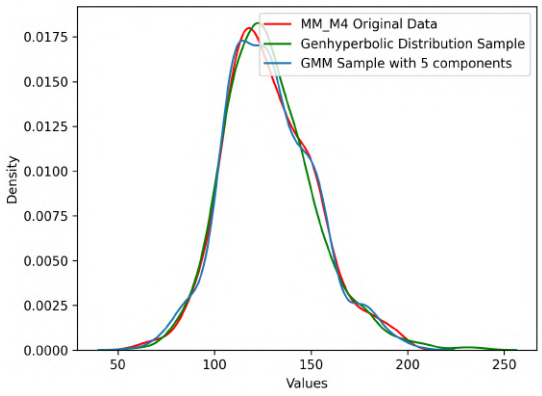
(a) GMM com 2 componentes



(b) GMM com 3 componentes

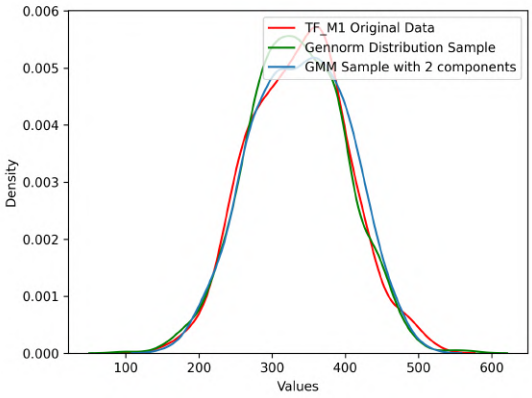


(c) GMM com 4 componentes

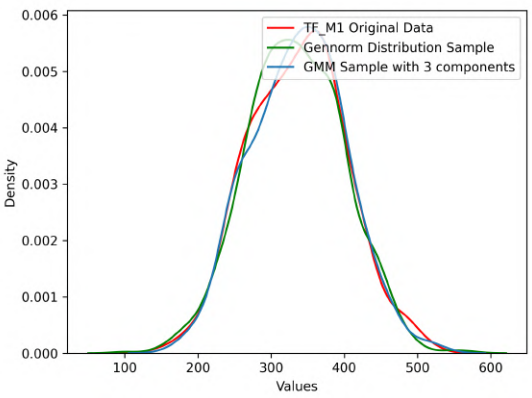


(d) GMM com 5 componentes

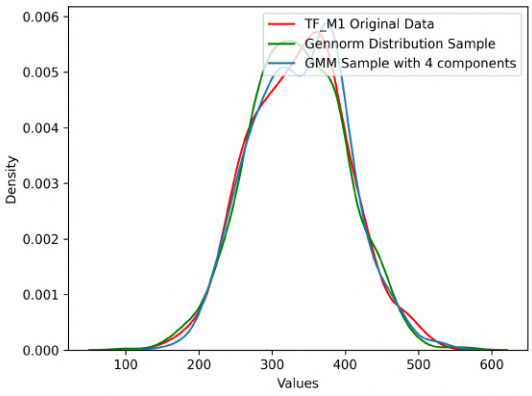
Figura B.45: Comparação das Estimativas KDE das distribuições ajustadas: turno MM_M4



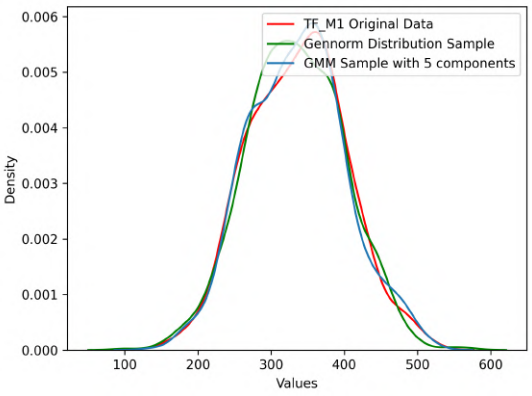
(a) GMM com 2 componentes



(b) GMM com 3 componentes

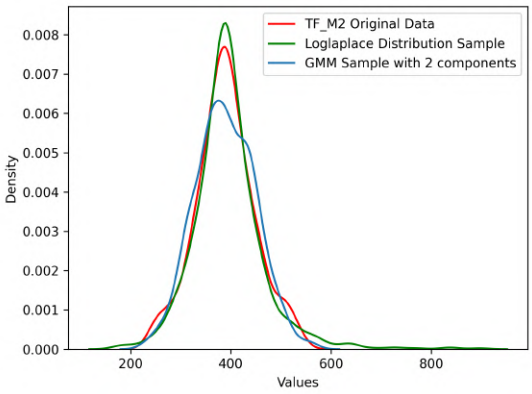


(c) GMM com 4 componentes

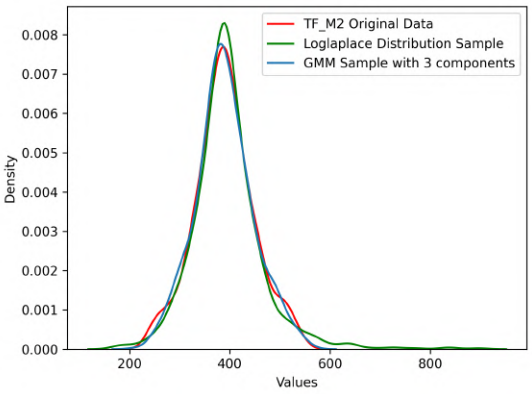


(d) GMM com 5 componentes

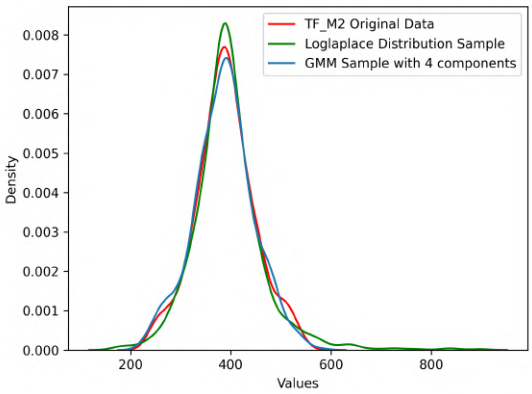
Figura B.46: Comparação das Estimativas KDE das distribuições ajustadas: turno TF_M1



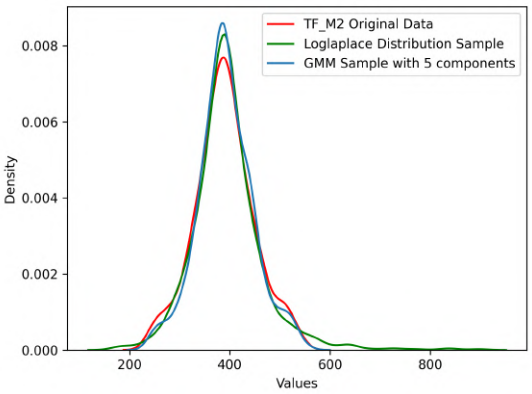
(a) GMM com 2 componentes



(b) GMM com 3 componentes

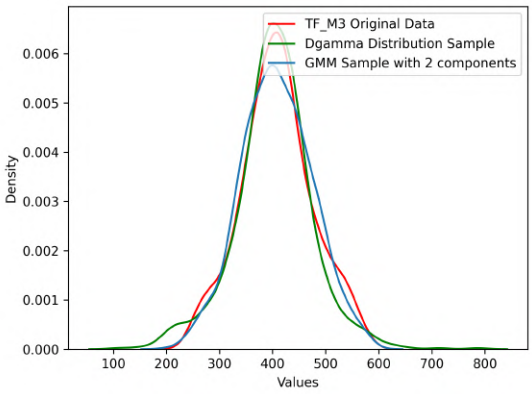


(c) GMM com 4 componentes

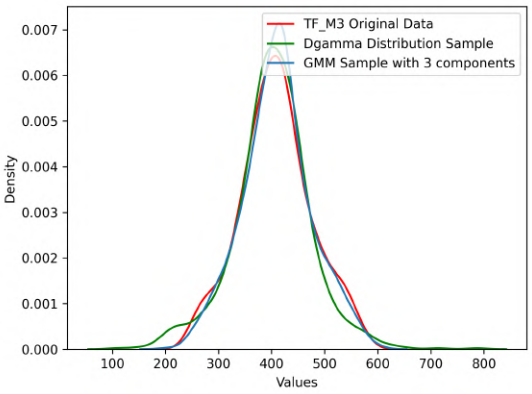


(d) GMM com 5 componentes

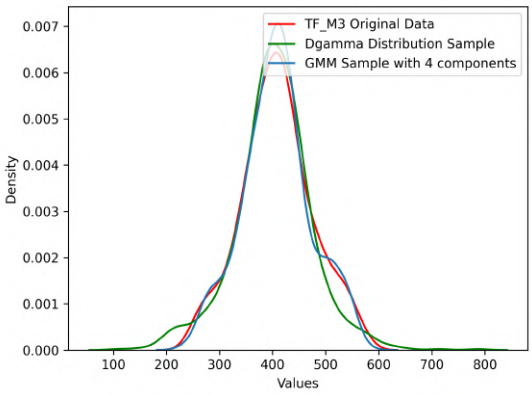
Figura B.47: Comparação das Estimativas KDE das distribuições ajustadas: turno TF_M2



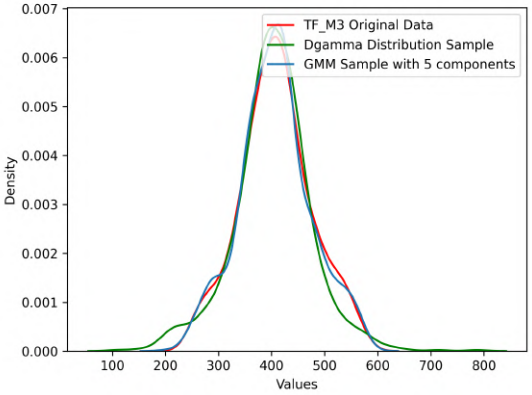
(a) GMM com 2 componentes



(b) GMM com 3 componentes

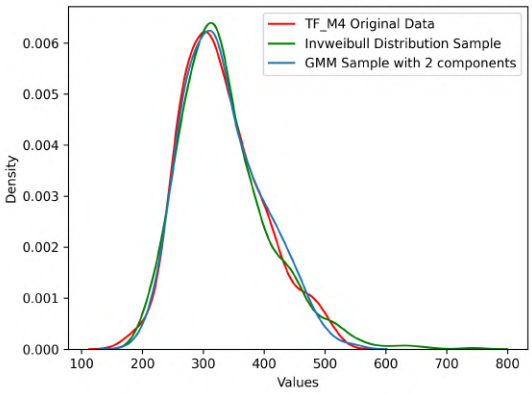


(c) GMM com 4 componentes

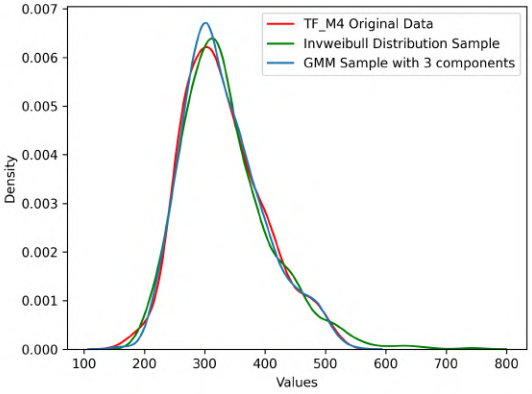


(d) GMM com 5 componentes

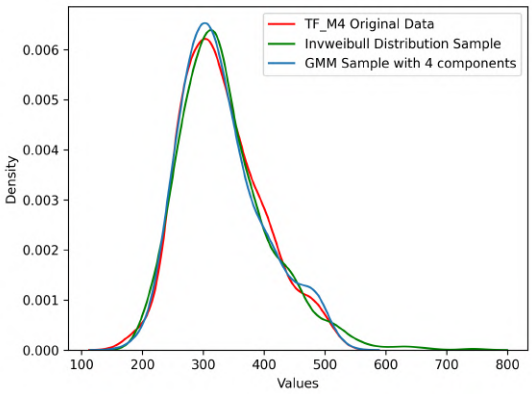
Figura B.48: Comparação das Estimativas KDE das distribuições ajustadas: turno TF_M3



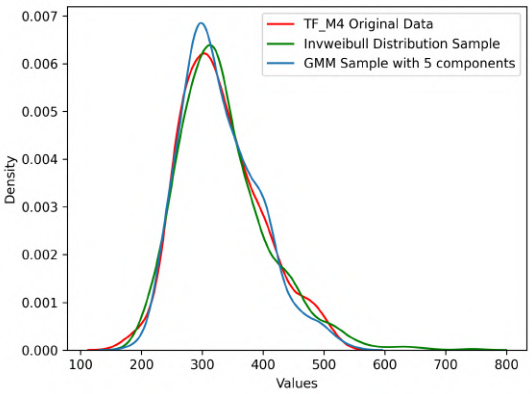
(a) GMM com 2 componentes



(b) GMM com 3 componentes



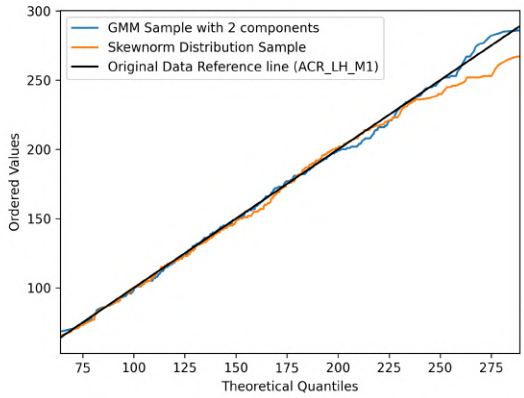
(c) GMM com 4 componentes



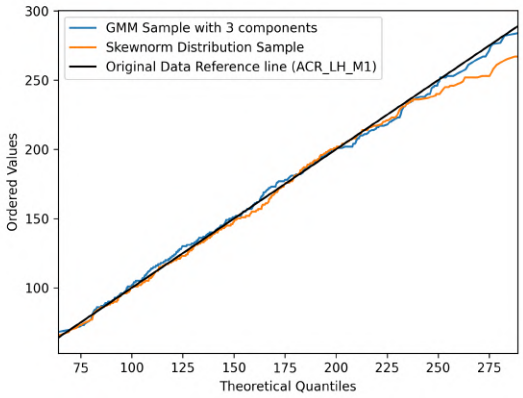
(d) GMM com 5 componentes

Figura B.49: Comparação das Estimativas KDE das distribuições ajustadas: turno TF_M4

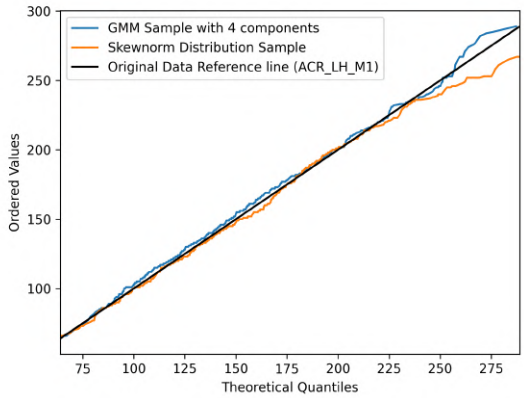
B.4 Gráficos *Q-Q plot* das amostras das distribuições ajustadas



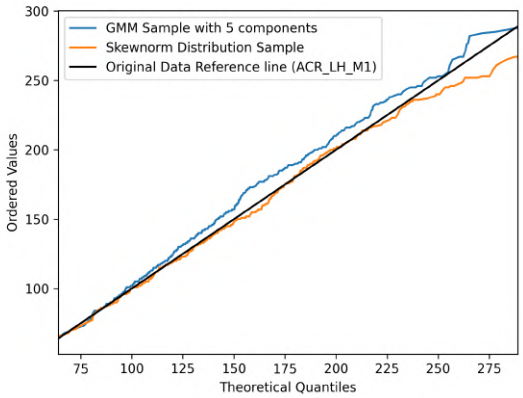
(a) GMM com 2 componentes



(b) GMM com 3 componentes

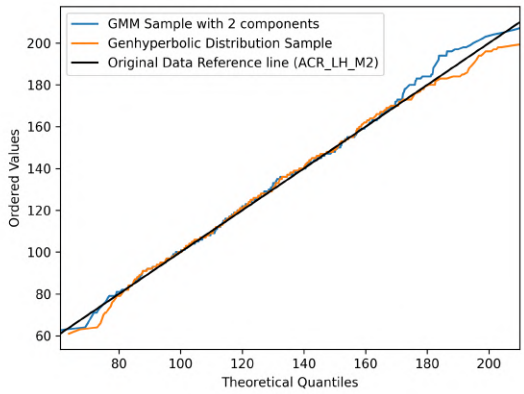


(c) GMM com 4 componentes

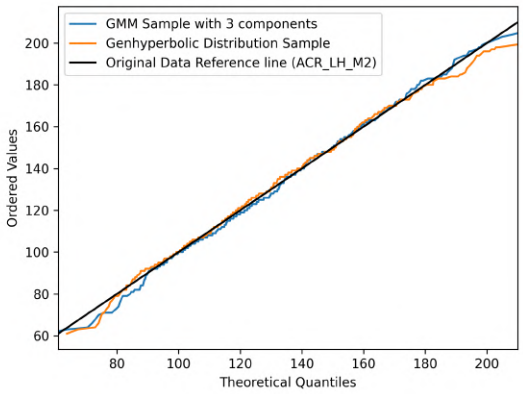


(d) GMM com 5 componentes

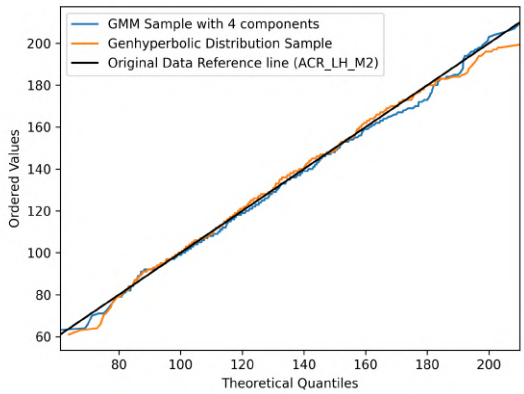
Figura B.50: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_LH_M1



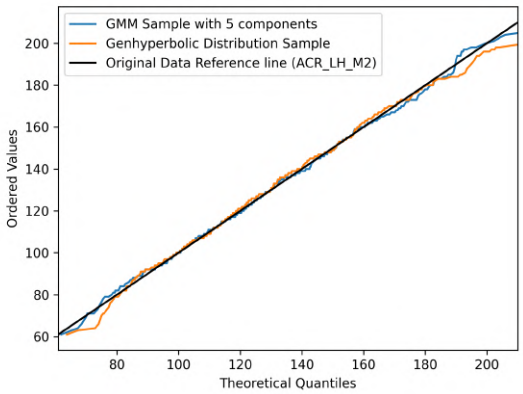
(a) GMM com 2 componentes



(b) GMM com 3 componentes

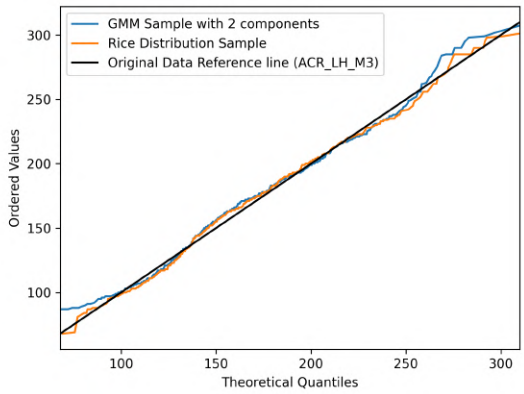


(c) GMM com 4 componentes

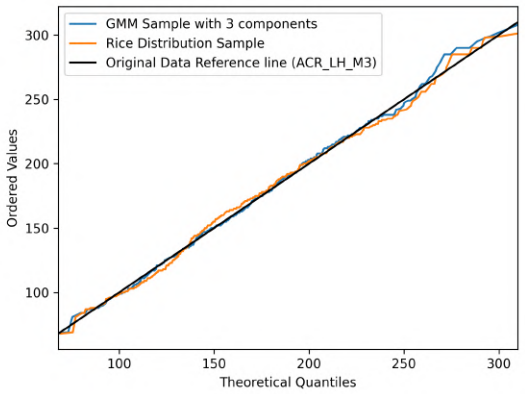


(d) GMM com 5 componentes

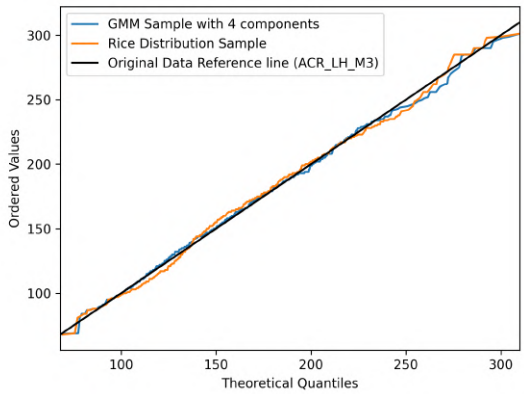
Figura B.51: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_LH_M2



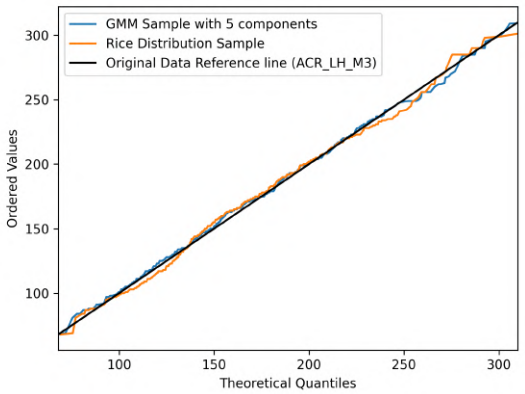
(a) GMM com 2 componentes



(b) GMM com 3 componentes

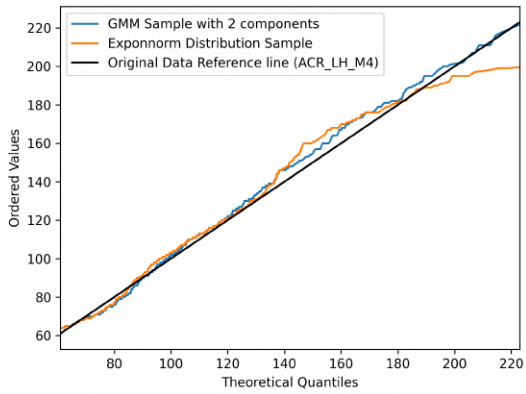


(c) GMM com 4 componentes

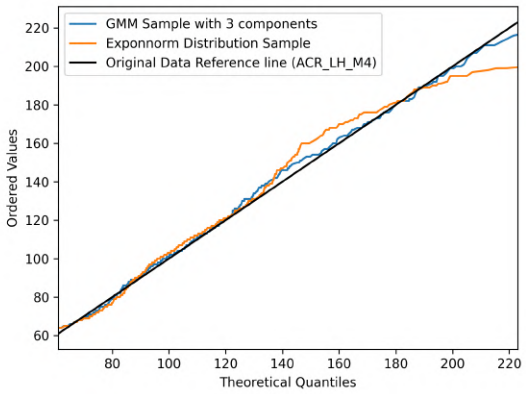


(d) GMM com 5 componentes

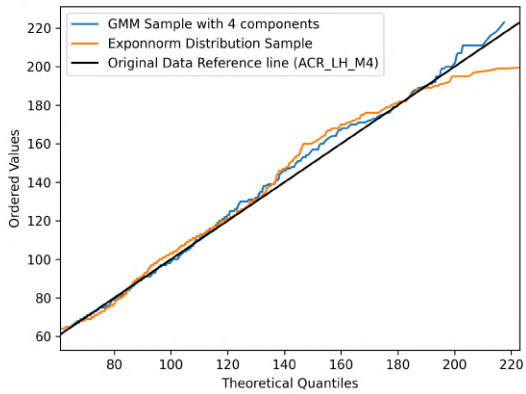
Figura B.52: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_LH_M3



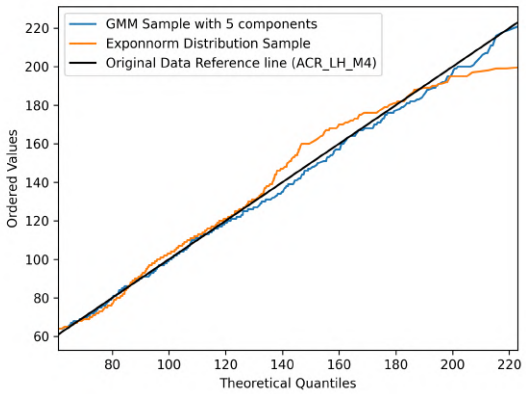
(a) GMM com 2 componentes



(b) GMM com 3 componentes

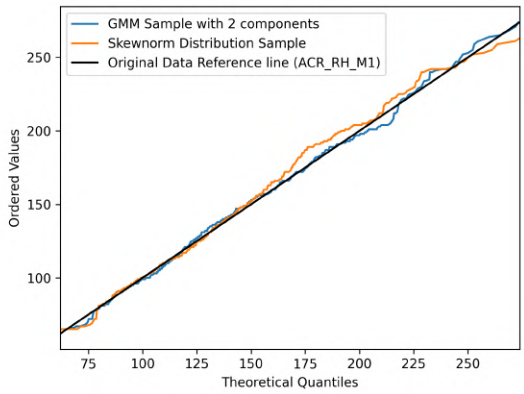


(c) GMM com 4 componentes

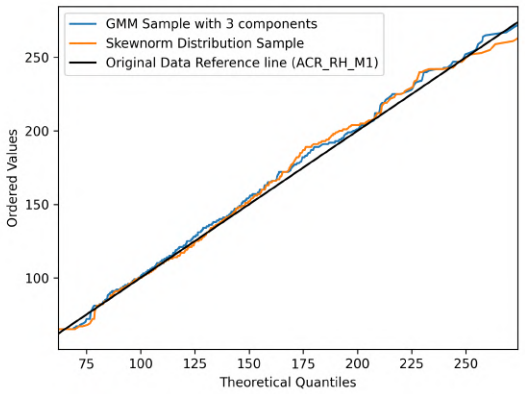


(d) GMM com 5 componentes

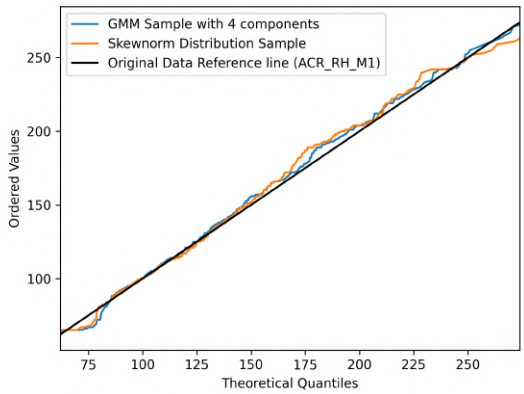
Figura B.53: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_LH_M4



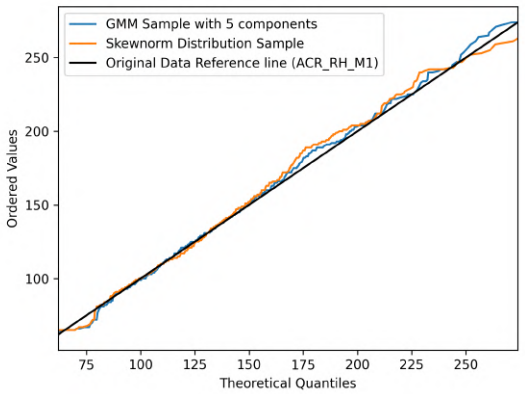
(a) GMM com 2 componentes



(b) GMM com 3 componentes

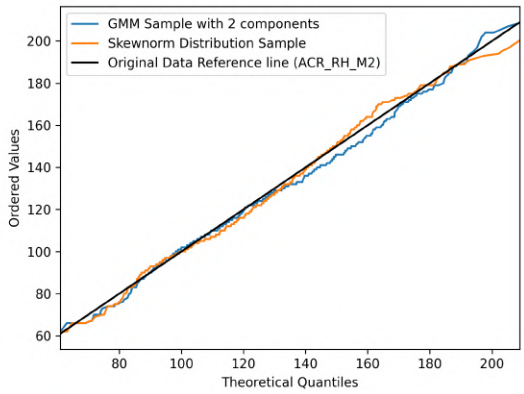


(c) GMM com 4 componentes

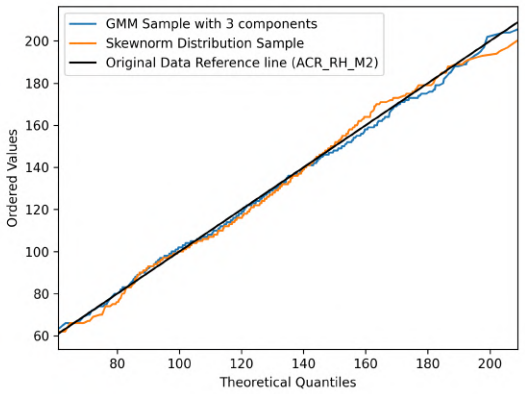


(d) GMM com 5 componentes

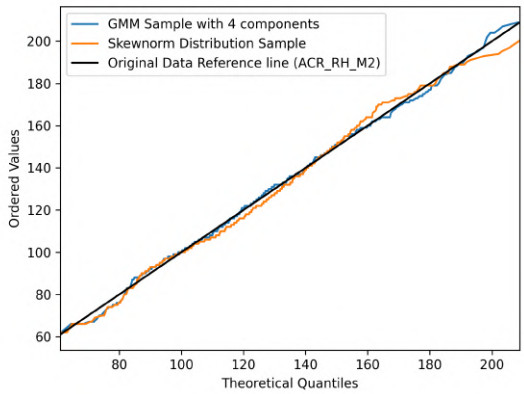
Figura B.54: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_RH_M1



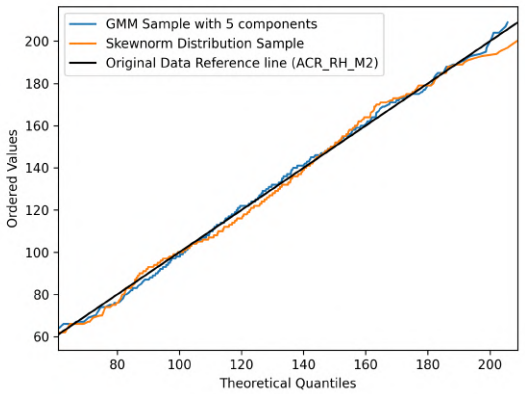
(a) GMM com 2 componentes



(b) GMM com 3 componentes

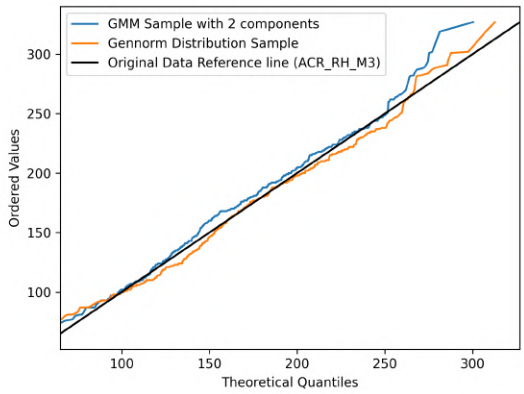


(c) GMM com 4 componentes

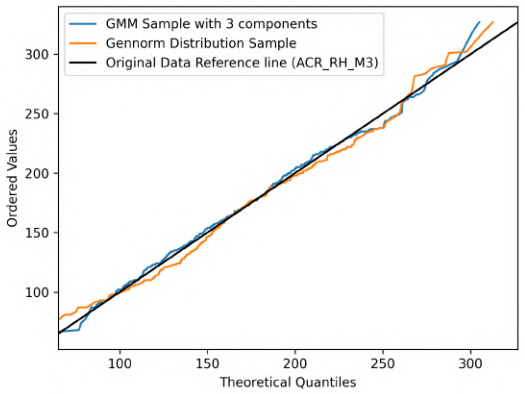


(d) GMM com 5 componentes

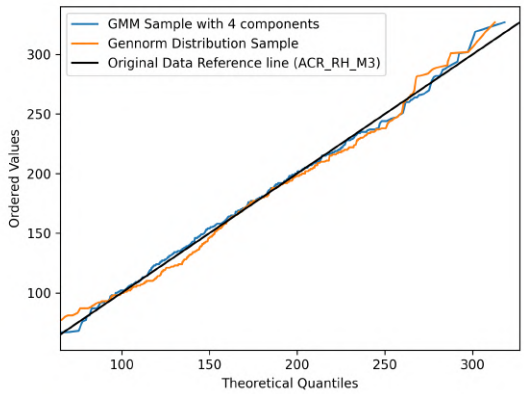
Figura B.55: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_RH_M2



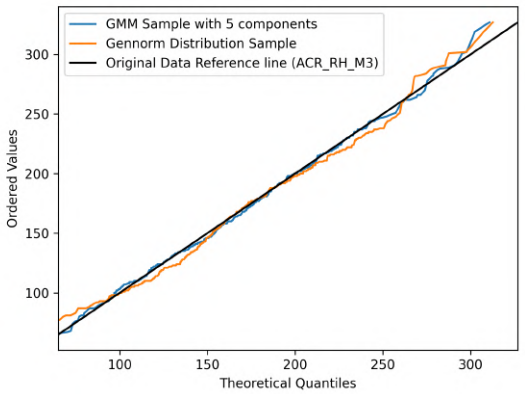
(a) GMM com 2 componentes



(b) GMM com 3 componentes

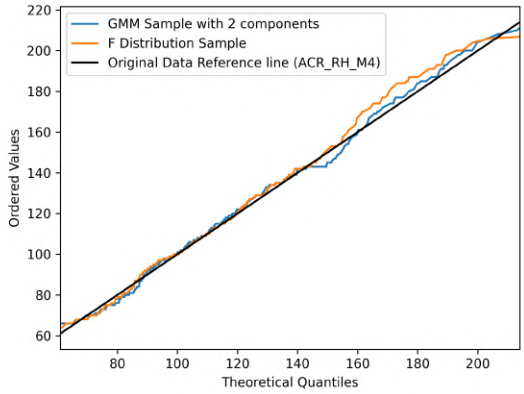


(c) GMM com 4 componentes

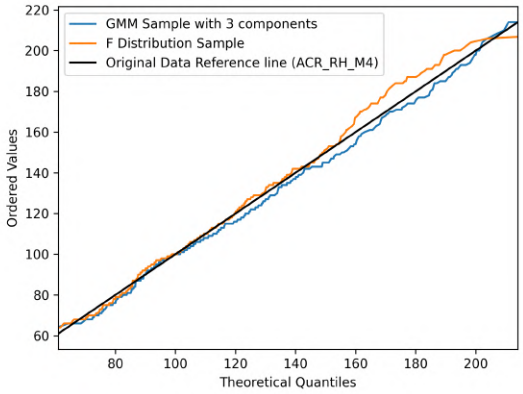


(d) GMM com 5 componentes

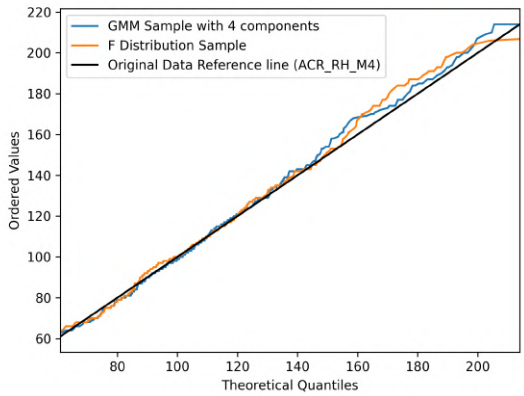
Figura B.56: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_RH_M3



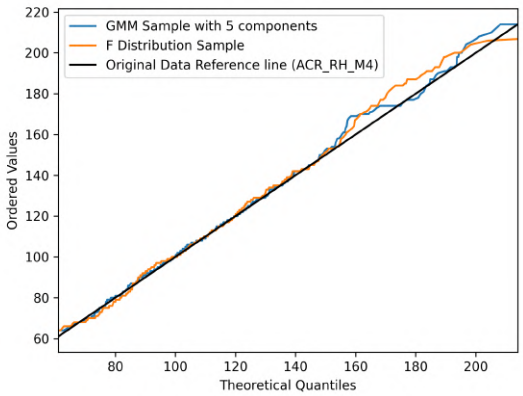
(a) GMM com 2 componentes



(b) GMM com 3 componentes

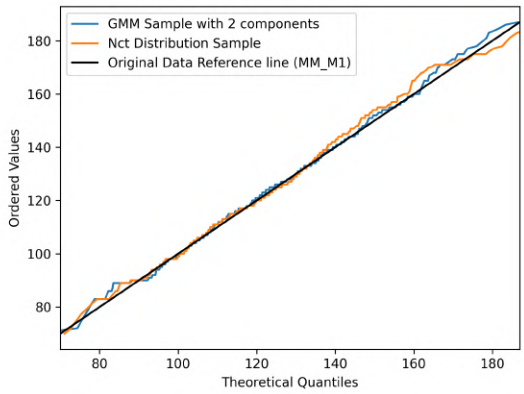


(c) GMM com 4 componentes

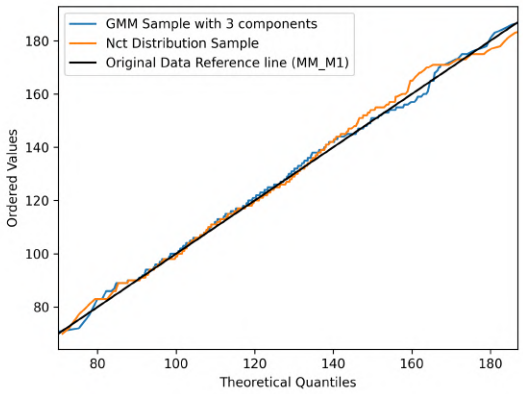


(d) GMM com 5 componentes

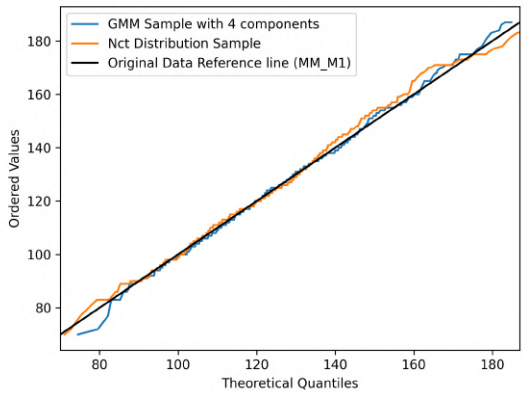
Figura B.57: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno ACR_RH_M4



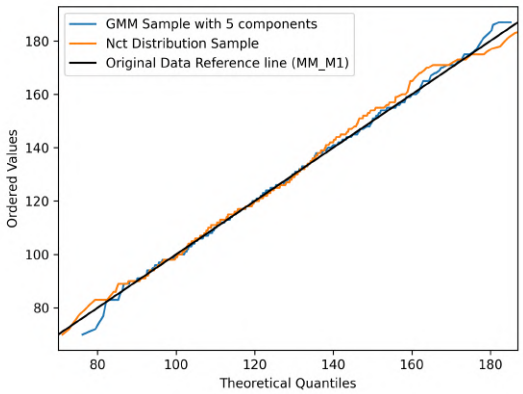
(a) GMM com 2 componentes



(b) GMM com 3 componentes

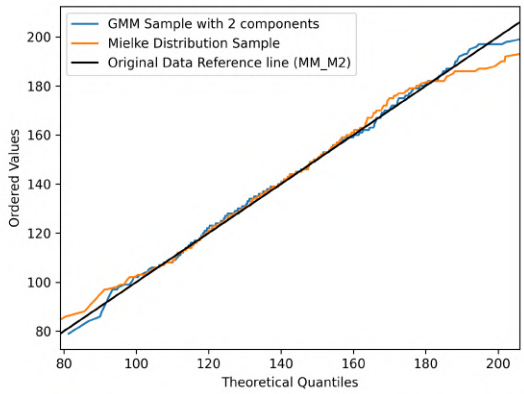


(c) GMM com 4 componentes

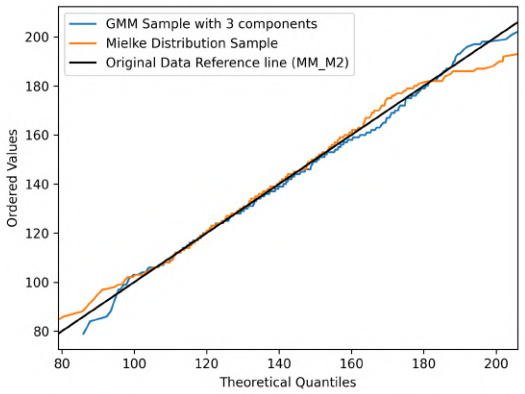


(d) GMM com 5 componentes

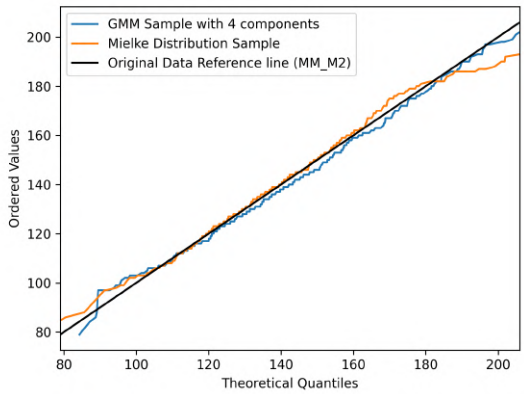
Figura B.58: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno MM_M1



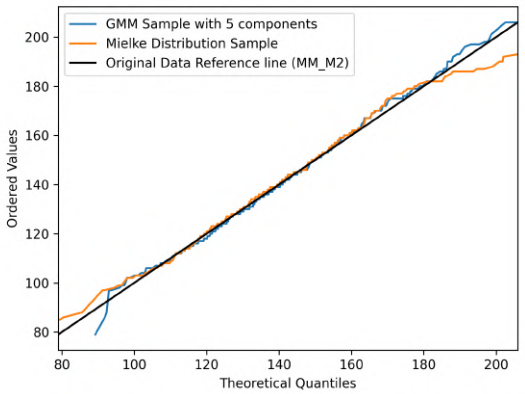
(a) GMM com 2 componentes



(b) GMM com 3 componentes

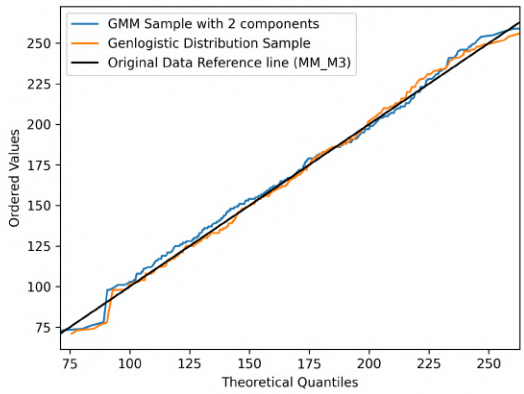


(c) GMM com 4 componentes

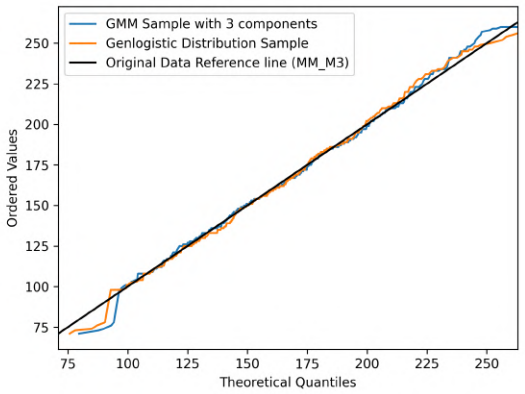


(d) GMM com 5 componentes

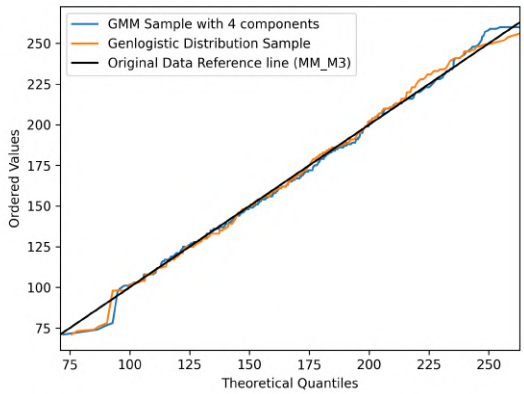
Figura B.59: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno MM_M2



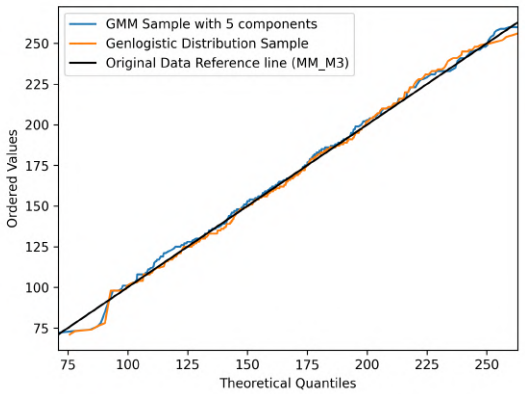
(a) GMM com 2 componentes



(b) GMM com 3 componentes

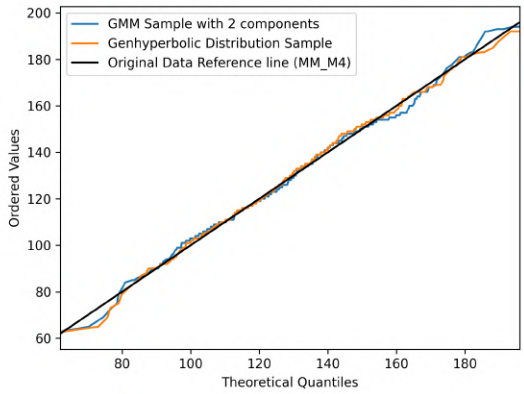


(c) GMM com 4 componentes

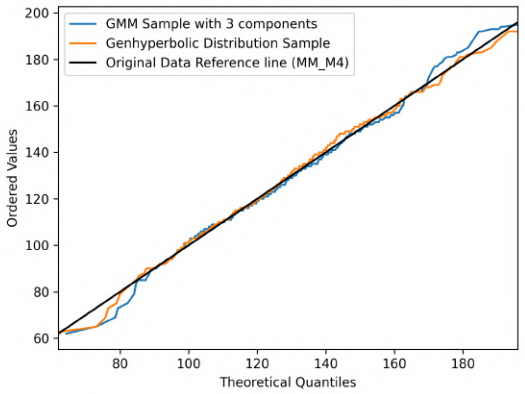


(d) GMM com 5 componentes

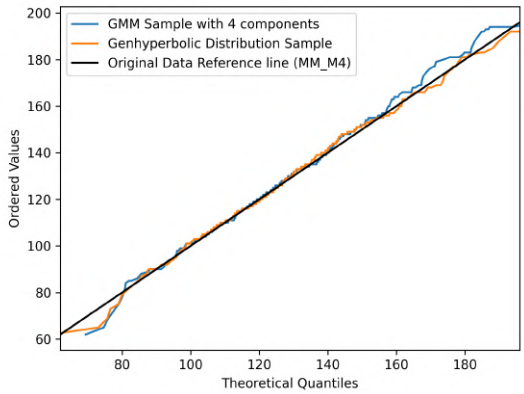
Figura B.60: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno MM_M3



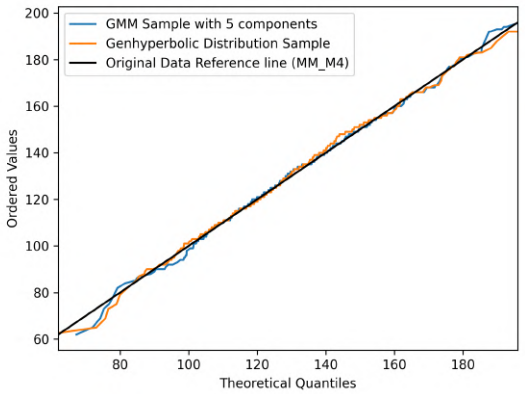
(a) GMM com 2 componentes



(b) GMM com 3 componentes

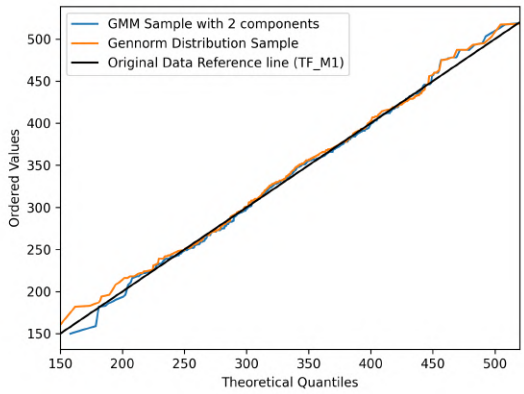


(c) GMM com 4 componentes

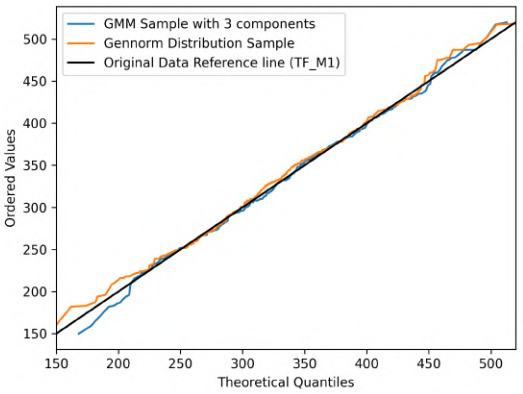


(d) GMM com 5 componentes

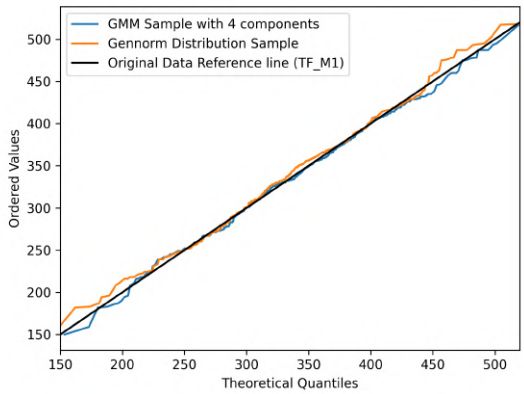
Figura B.61: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno MM_M4



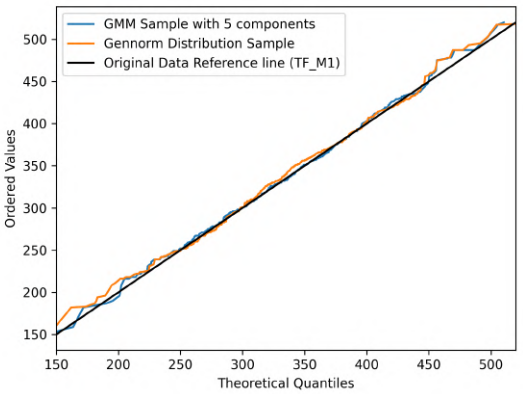
(a) GMM com 2 componentes



(b) GMM com 3 componentes

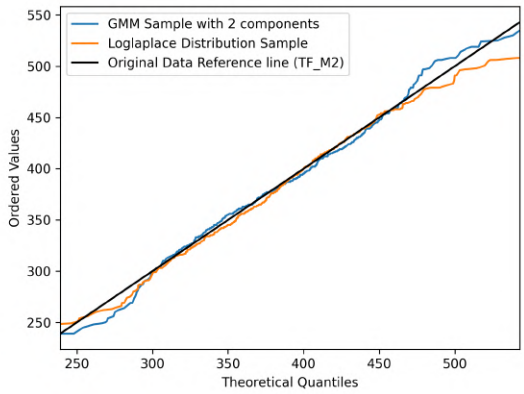


(c) GMM com 4 componentes

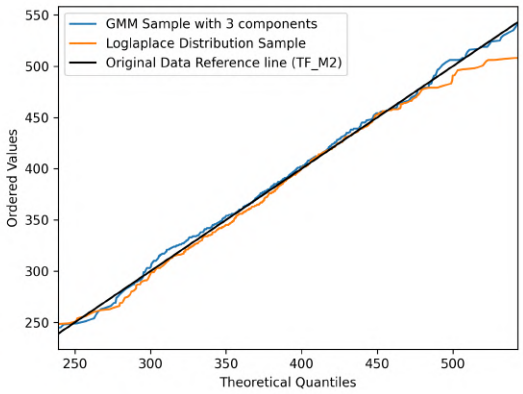


(d) GMM com 5 componentes

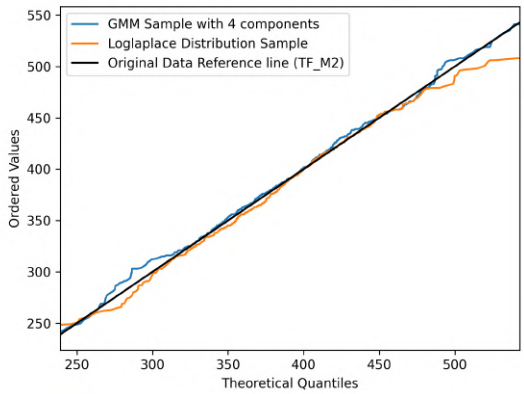
Figura B.62: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno TF_M1



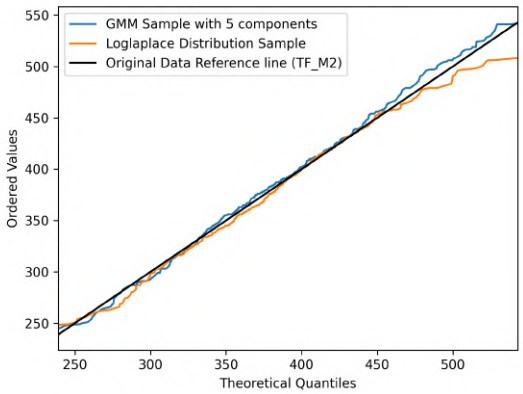
(a) GMM com 2 componentes



(b) GMM com 3 componentes

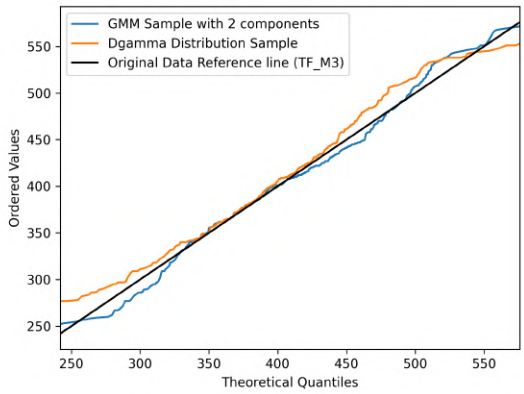


(c) GMM com 4 componentes

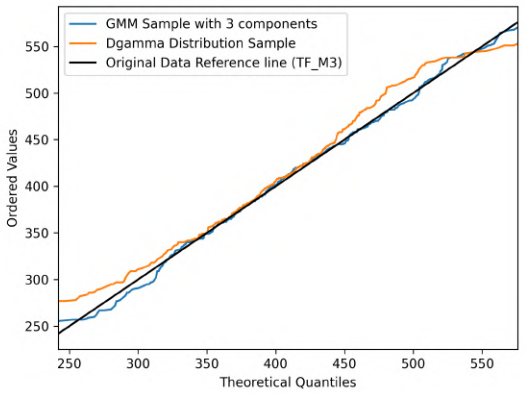


(d) GMM com 5 componentes

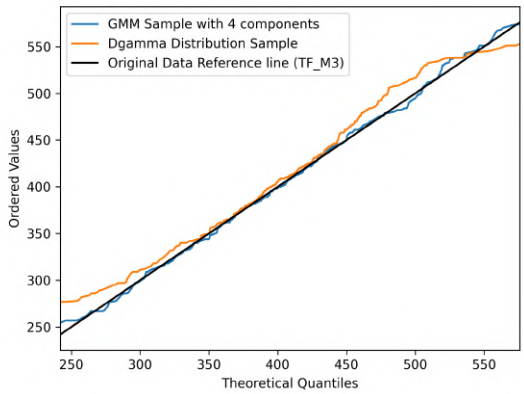
Figura B.63: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno TF_M2



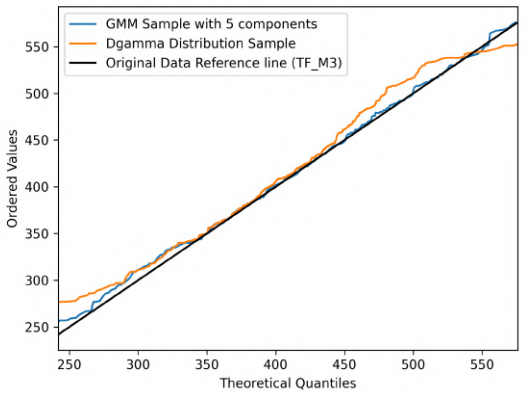
(a) GMM com 2 componentes



(b) GMM com 3 componentes

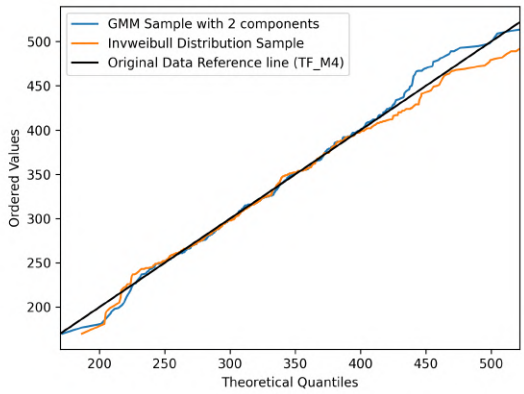


(c) GMM com 4 componentes

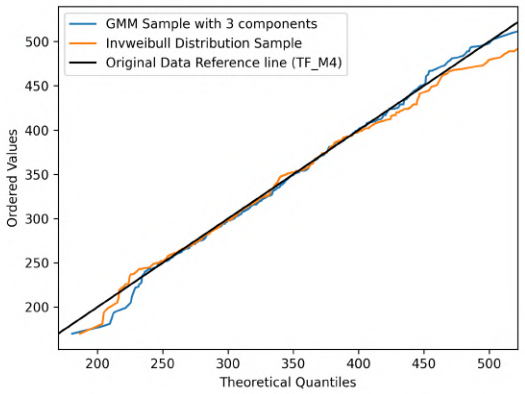


(d) GMM com 5 componentes

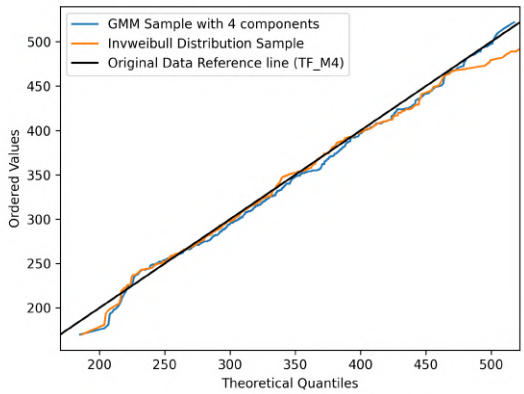
Figura B.64: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno TF_M3



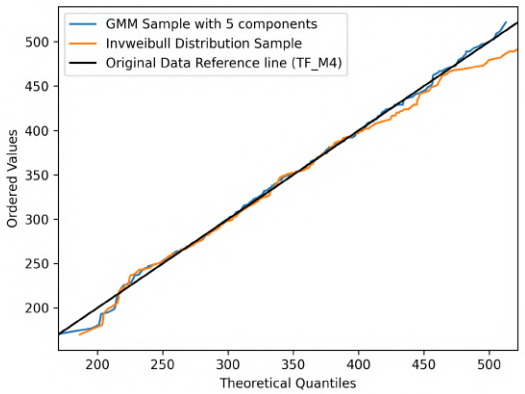
(a) GMM com 2 componentes



(b) GMM com 3 componentes



(c) GMM com 4 componentes



(d) GMM com 5 componentes

Figura B.65: Comparação dos *Q-Q Plot* das distribuições ajustadas: turno TF_M4

B.5 Testes e indicadores de ajustamento

Tabela B.2: Resultado dos testes de ajustamento dos tempos de *picking* . A coluna $l(\boldsymbol{\theta}_{MLE})$ refere-se ao indicador *Log-Likelihood*. A coluna **Stat** refere-se à estatística de teste obtida. Linhas preenchidas a azul correspondem ás distribuições escolhidas para os turnos do cenário pessimista. Linhas preenchidas a verde correspondem ás distribuições escolhidas para os turnos do cenário otimista

Testes e Indicadores de ajustamento								
			AD		KS			
Turnos	**Distribuição**	$l(\boldsymbol{\theta}_{MLE})$	**Stat**	**p-value**	**Stat**	**p-value**	**BIC**	**AIC**
ACR_LH_M1	Normal Assimétrica	-2753	-0.5396	≥0,25	0.0413	0.5810	5525	5513
	GMM							
	n=2	-2748	-0.8811	≥0,25	0.0325	0.8439	5527	5506
	n=3	-2748	-0.5411	≥0,25	0.0453	0.4630	5546	5512
	n=4	-2745	-0.3045	≥0,25	0.0463	0.4353	5558	5511
	n=5	-2744	2.9890	0.0197	0.0761	0.0344	5575	5516
ACR_LH_M2	Hiperbólica Genérica	-2464	-0.1534	≥0,25	0.0476	0.4086	4958	4937
	GMM							
	n=2	-2459	-0.6644	≥0,25	0.0371	0.7191	4950	4929
	n=3	-2458	-0.8747	≥0,25	0.0293	0.9208	4965	4931
	n=4	-2458	-0.8190	≥0,25	0.0357	0.7610	4985	4938
	n=5	-2455	-0.3359	≥0,25	0.0437	0.5156	5000	4937
ACR_LH_M3	Rice	-2541	-0.6476	0.1785	0.0585	0.2057	5100	5087
	GMM							
	n=2	-2547	0.4839	0.2098	0.0700	0.0778	5125	5104
	n=3	-2536	-0.7523	≥0,25	0.0370	0.7458	5121	5088
	n=4	-2531	-0.6739	≥0,25	0.0489	0.4158	5130	5084
	n=5	-2530	-0.4835	≥0,25	0.0475	0.4386	5146	5087
ACR_LH_M4	ExponNorm*	-2378	0.9684	0.1304	0.0604	0.1784	4774	4761
	GMM							
	n=2	-2357	0.1526	≥0,25	0.0551	0.2673	4745	4725
	n=3	-2354	-0.8192	≥0,25	0.0342	0.8256	4756	4723
	n=4	-2348	-0.5897	≥0,25	0.0429	0.5704	4764	4718
	n=5	-2346	-0.8193	≥0,25	0.0340	0.8323	4779	4721
ACR_RH_M1	Normal Assimétrica	-2629	-0.1766	≥0,25	0.0556	0.2379	5276	5263
	GMM							
	n=2	-2627	-0.7126	≥0,25	0.0390	0.6673	5286	5265
	n=3	-2625	-0.3205	≥0,25	0.0417	0.5835	5300	5266
	n=4	-2621	-0.4244	≥0,25	0.0436	0.5259	5310	5263
	n=5	-2620	-0.8571	≥0,25	0.0265	0.9661	5327	5268
ACR_RH_M2	Normal Assimétrica	-2452	0.4684	0.2131	0.0628	0.1296	4923	4910
	GMM							
	n=2	-2449	-0.0056	≥0,25	0.0521	0.3017	4928	4907
	n=3	-2450	-0.4951	≥0,25	0.0401	0.6271	4949	4915
	n=4	-2448	-0.9017	≥0,25	0.0308	0.8903	4965	4918
	n=5	-2445	-0.7042	≥0,25	0.0354	0.7687	4977	4918

Continua na página seguinte

Testes e Indicadores de ajustamento								
			AD		KS			
Turnos	Distribuição	$l(\theta_{MLE})$	Stat	p-value	Stat	p-value	BIC	AI
ACR_RH_M3	Normal Generalizada	-2555	0.9760	0.1294	0.0663	0.1105	5128	5115
	GMM							
	n=2	-2542	0.6050	0.1862	0.0621	0.1560	5115	5094
	n=3	-2538	-0.6812	≥0,25	0.0392	0.6845	5126	5092
	n=4	-2535	-0.7834	≥0,25	0.0427	0.5779	5138	5092
	n=5	-2535	-0.9368	≥0,25	0.0304	0.9142	5156	5098
ACR_RH_M4	Fisher	-2128	-0.3174	≥0,25	0.0433	0.5988	4280	4263
	GMM							
	n=2	-2126	-0.3890	≥0,25	0.0436	0.5929	4281	4261
	n=3	-2120	0.2368	≥0,25	0.0486	0.4534	4288	4255
	n=4	-2110	-0.6512	≥0,25	0.0362	0.8043	4288	4243
	n=5	-2107	-1.0247	≥0,25	0.0313	0.9150	4299	4242
MM_M1	NCT**	-1873	-0.5937	≥0,25	0.0393	0.7374	3771	3755
	GMM							
	n=2	-1871	-0.9002	≥0,25	0.0388	0.7526	3772	3752
	n=3	-1870	-0.6498	≥0,25	0.0414	0.6782	3788	3756
	n=4	-1867	-1.0147	≥0,25	0.0306	0.9364	3800	3756
	n=5	-1867	-1.0896	≥0,25	0.0260	0.9855	3819	3762
MM_M2	Mielke	-2261	-0.5741	≥0,25	0.0412	0.6070	4547	4530
	GMM							
	n=2	-2251	-0.4025	≥0,25	0.0514	0.3282	4534	4513
	n=3	-2251	-0.6346	≥0,25	0.0441	0.5177	4552	4518
	n=4	-2251	0.8561	0.1455	0.0704	0.0694	4571	4525
	n=5	-2249	-0.8324	≥0,25	0.0355	0.7768	4585	4526
MM_M3	Logística Genérica	-2291	-0.6339	≥0,25	0.0420	0.6208	4600	4587
	GMM							
	n=2	-2288	0.0329	≥0,25	0.0595	0.2082	4606	4585
	n=3	-2286	-0.8477	≥0,25	0.0352	0.8166	4621	4588
	n=4	-2284	-0.4586	≥0,25	0.0428	0.5954	4636	4590
	n=5	-2280	-0.6547	≥0,25	0.0479	0.4535	4646	4588
MM_M4	Hiperbólica Genérica	-2282	-0.7053	≥0,25	0.0431	0.5454	4596	4575
	GMM							
	n=2	-2283	-0.5282	≥0,25	0.0404	0.6276	4598	4577
	n=3	-2284	-0.4367	≥0,25	0.0431	0.5441	4618	4584
	n=4	-2276	-0.7078	≥0,25	0.0411	0.6048	4620	4574
	n=5	-2275	-0.9257	≥0,25	0.0414	0.5966	4637	4578
TF_M1	Normal Generalizada	-2874	-0.4613	≥0,25	0.0492	0.3690	5767	5754
	GMM							
	n=2	-2875	-0.7868	≥0,25	0.0340	0.8101	5781	5759
	n=3	-2873	-0.8579	≥0,25	0.0272	0.9556	5796	5762
	n=4	-2873	-0.8429	≥0,25	0.0346	0.7939	5815	5768
	n=5	-2871	-0.9905	≥0,25	0.0277	0.9487	5829	5769

Continua na página seguinte

Testes e Indicadores de ajustamento								
			AD		KS			
Turnos	**Distribuição**	$l(\theta_{MLE})$	**Stat**	**p-value**	**Stat**	**p-value**	**BIC**	**AI**
TF_M2	Log-Laplace	-2734	-0.0241	≥0,25	0.0431	0.5505	5487	5475
	GMM							
	n=2	-2730	-0.0967	≥0,25	0.0524	0.3066	5491	5470
	n=3	-2721	-0.7132	≥0,25	0.0345	0.8075	5491	5457
	n=4	-2719	-0.8695	≥0,25	0.0280	0.9476	5506	5459
	n=5	-2714	-0.5528	≥0,25	0.0397	0.6521	5515	5456
TF_M3	Gamma Dupla	-2588	1.0754	0.1175	0.0562	0.2607	5195	5182
	GMM							
	n=2	-2584	0.2549	≥0,25	0.0585	0.2204	5199	5178
	n=3	-2575	-0.8694	≥0,25	0.0376	0.7461	5199	5166
	n=4	-2572	-1.0192	≥0,25	0.0259	0.9797	5211	5165
	n=5	-2569	-0.9894	≥0,25	0.0276	0.9630	5225	5167
TF_M4	Weibull Invertida	-2873	-0.4042	≥0,25	0.0324	0.8543	5765	5753
	GMM							
	n=2	-2863	-0.6609	≥0,25	0.0284	0.9377	5756	5735
	n=3	-2863	-0.5714	≥0,25	0.0426	0.5518	5777	5743
	n=4	-2860	0.6241	0.1827	0.0556	0.2346	5789	5743
	n=5	-2860	-0.9115	≥0,25	0.0251	0.9781	5807	5748

* A distribuição ExponNorm tem o nome de Distribuição Normal modificada exponencialmente.

** A distribuição NCT tem o nome de Distribuição t de Student descentralizada.

APÊNDICE C

ANÁLISE DOS TEMPOS DE *LOAD/UNLOAD*

C.1 Gráficos *box-plot* para deteção de *outliers*

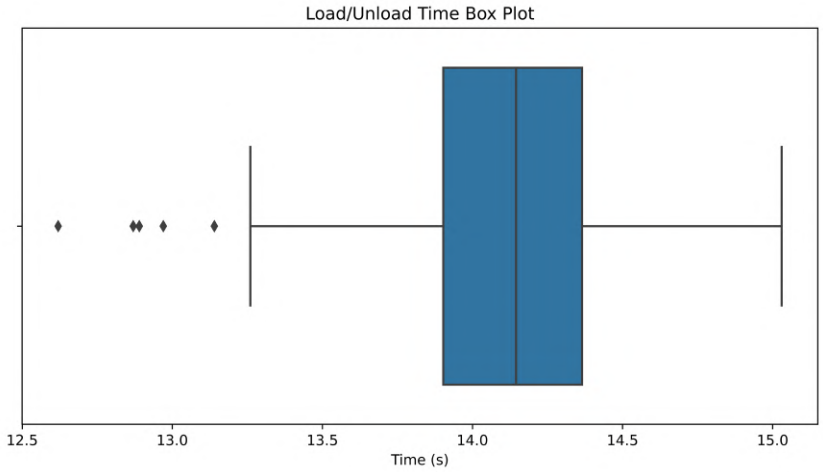


Figura C.1: Gráfico *box-plot*: Tempos de *load/unload*

Tabela C.1: Média e desvio padrão dos tempos de *load/unload* dos *Autonomous Mobile Vehicles* ou Veículos Moveis Autónomos (AMR), após deteção e eliminação de *outliers*. A coluna n indica a dimensão da amostra inicial, antes da identificação de *outliers*. A coluna n_f indica a dimensão da amostra, após a identificação e eliminação de *outliers*

Dados Originais (sem *outliers*)						
Dados	n	n_f	**Média (s)**	**Desvio padrão (s)**	**Mínimo (s)**	**Máximo (s)**
Load/Unload	150	144	14.13	0.36	13.26	15.03

C.2 Histograma e estimativa KDE dos tempos de *load/unload*

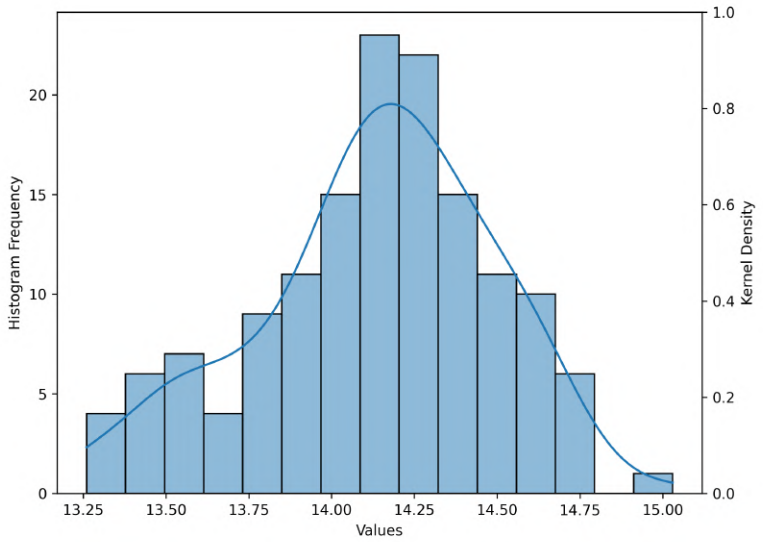
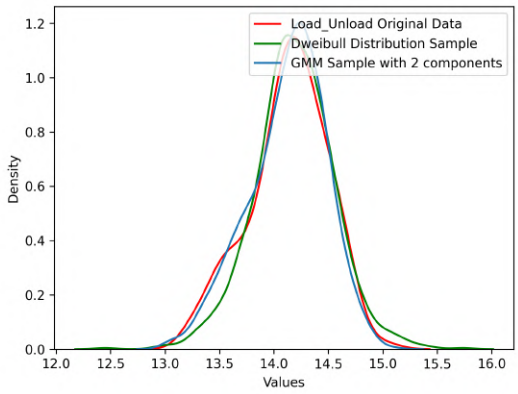
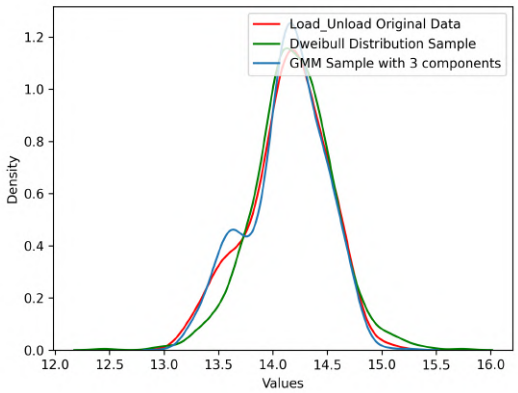


Figura C.2: Histograma e respetiva KDE dos dados originais sem *outliers*: Tempos de *load/unload*

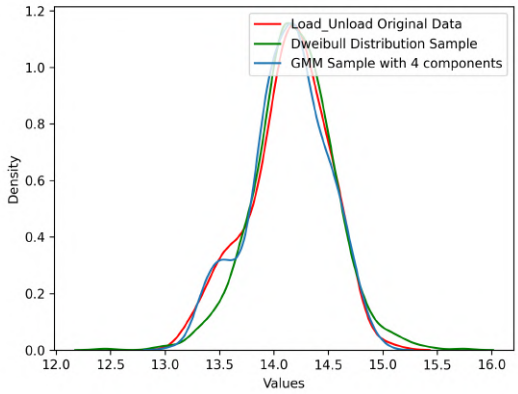
C.3 Estimativas KDE das amostras das distribuições ajustadas



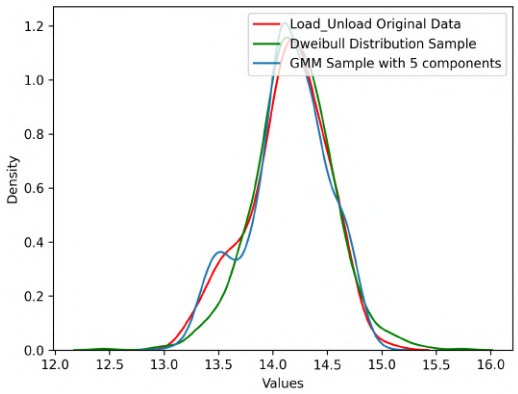
(a) *Gaussian Mixture Model* ou Modelo de Mistura Gaussiana (GMM) com 2 componentes



(b) GMM com 3 componentes



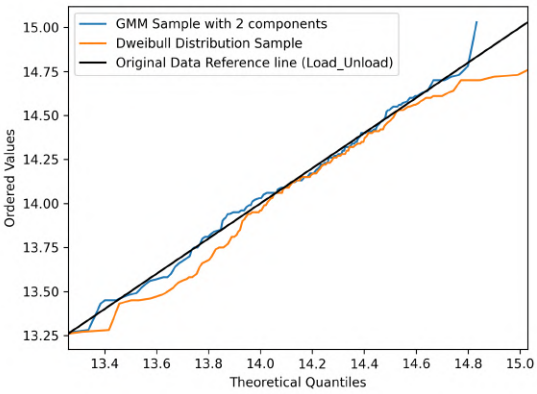
(c) GMM com 4 componentes



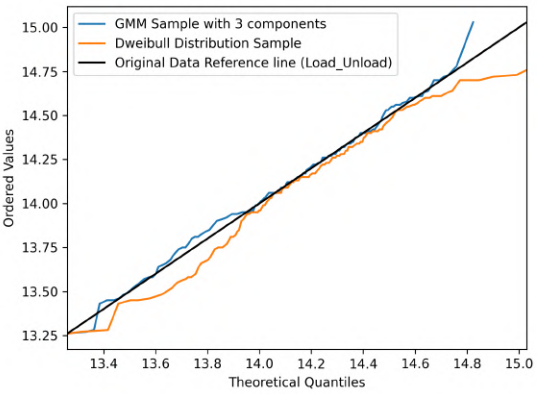
(d) GMM com 5 componentes

Figura C.3: Estimativas KDE dos dados originais e distribuições ajustadas pelo "Fitter"e GMM: Tempos de *load/unload*

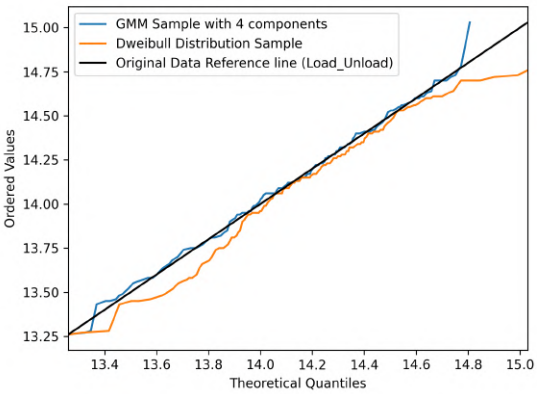
C.4 Gráficos *Q-Q plot* das amostras das distribuições ajustadas



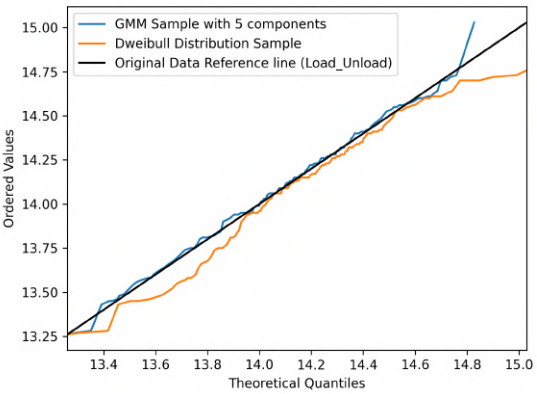
(a) GMM com 2 componentes



(b) GMM com 3 componentes



(c) GMM com 4 componentes



(d) GMM com 5 componentes

Figura C.4: Comparação dos *Q-Q Plot* das distribuições ajustadas: Tempos de *load/unload*

C.5 Testes e indicadores de ajustamento

Tabela C.2: Resultado dos testes de ajustamento dos tempos de *load/unload* . A coluna $l(\boldsymbol{\theta_{MLE}})$ refere-se ao indicador *Log-Likelihood*. A coluna **Stat** refere-se à estatística de teste obtida. A linha preenchida a verde corresponde à distribuição escolhida.

Testes e Indicadores de ajustamento								
			AD		**KS**			
Turnos	**Distribuição**	$l(\boldsymbol{\theta_{MLE}})$	**Stat**	**p-value**	**Stat**	**p-value**	**BIC**	**AIC**
Load/Unload	Weibull Dupla	-57.5	0.3746	0.2339	0.0651	0.6345	129,9	121.0
	GMM							
	n=2	-52.6	-0.9028	≥0,25	0.0537	0.8392	130,0	115,2
	n=3	-50.0	-1.1109	≥0,25	0.0377	0.9906	139.8	116.1
	n=4	-50.5	-1.1049	≥0,25	0.0397	0.9836	155.7	123.0
	n=5	-50.3	-1.1037	≥0,25	0.0304	0.9994	170.2	128.6

APÊNDICE D

Scripts Python do problema de Programação Linear Inteira Mista

File: Robust_Schedule_1.py

```python
from gurobipy import *
import gurobipy as grb
import numpy as np
import pandas as pd
import time as clock
import scipy.stats as ss
from datetime import datetime


# function to generate a sample from a GMM model with
# 2 components fitted to the load and unload times measured
from load_unload_sample import load_unload_sample


def robust_schedule(amr, task_list, time, last_tasktype, p, number):
    start_clock = clock.time()

    # get current date and time
    current_datetime = datetime.now().strftime("%Y-%m-%d %H-%M-%S")

    # convert datetime obj to string
    str_current_datetime = str(current_datetime)
```

Figura D.1: *Script Python* para a otimização do modelo MILP proposto (Parte 1).

File: Robust_Schedule_1.py

```
22
23      # ---------------------------------------- GUROBI PARAMETERS
    ----------------------------------------
24
25      # Create model to optimize
26      opt_model = grb.Model(name="MILP Model")
27
28      # Set the DualReductions parameter to 1 (on) or 0 (off)
29      opt_model.Params.DualReductions = 0  # Default = 1, with 1 (on) or 0 (off)
30
31      # When some variables are integers and their values, after optimization, are nearly integers
32      # Check the rounded solution for feasibility and perform additional branching (Branch and Cut
    Algorithm) steps to
33      # try to remove the integrality violation without compromising feasibility.
34      opt_model.Params.IntegralityFocus = 1  # Default = 0, with 1 (on) or 0 (off)
35      opt_model.Params.IntFeasTol = 1e-9  # Default = 1e-5, Minimum = 1e-9, Maximum = 1e-1
36
37      opt_model.Params.Aggregate = 0  # The options are off (0), moderate (1, Default), or aggressive (2)
38
39      opt_model.Params.Presolve = 1  # Presolve routines are enabled
40
41      # Ajust Gap parameter to the desired value
42      opt_model.Params.MIPGap = 0  # Default = 1e-4, Minimum = 0, Maximum = Infinity
43
44      # Set the output to print to console parameter to 1 (on) or 0 (off)
45      opt_model.Params.LogToConsole = 1  # Default = 1, with 1 (on) or 0 (off)
46      opt_model.Params.OutputFlag = 1  # Default = 1, with 1 (on) or 0 (off)
47
48      # --------------------- MODEL PARAMETERS ---------------------
49
50      # -------------------- GENERAL PARAMETERS ---------------------
51
52      m = len(amr)  # Number of AMR (Autonomous Mobile Robots)
53      n = len(task_list[0])  # Number of tasks to schedule
54
55      # ----------------------- MODEL INDEXES -----------------------
56
57      # Index Initialization
58      i: int = 0
59      j: int = 0
60      k: int = 0
61      l: int = 0
62
63      set_i = range(0, m)  # M set
64
65      set_j = range(0, n)  # J set
66
67      set_k = range(0, n)  # J set
68
69      set_l = range(-1, n)  # L set
70
71      # --------------------- RACK PARAMETERS ----------------------
72
73      task_ID = [int(i) for i in task_list[0]]
74      # task_ID, so that, when a schedule is generated, the robot knows the which task to pull during the
    simulation
75
76      deadline = [i for i in task_list[3]]
77      # instant when the task ends (rack is free and, in some cases, the production line has zero parts)
78
79      delay = [int(i) for i in task_list[2]]
80      # delay necessary in some tasks
81
82      time = [float(i) for i in time]
83
84      # --------------------- AMR PARAMETERS ----------------------
85
86      # Load and unload distributions, using an empirical distribution from a GMM model with 2 components
87      load = [[round(load_unload_sample(1)[0], ndigits=2) for j in set_j] for i in set_i]  # AMR load time
88      unload = [[round(load_unload_sample(1)[0], ndigits=2) for j in set_j] for i in set_i]  # AMR unload
    time
```

Figura D.2: *Script Python* para a otimização do modelo MILP proposto (Parte 2).

File: Robust_Schedule_1.py

```
89
90      # print("\n", "load = ", load)
91      # print("unload = ", unload)
92
93      task_type = [int(i) for i in task_list[1]]
94
95      # task_type[j] =    0      1       2       3       4       5       6       7
96      #           b =    0      1       2       3       4       5       6       7
97      t_c = np.array([[0     , 60.96 , 20.47, 195.69, 207.91, 89.63, 122.14, 162.67],  # a = 0 ,
    task_type[l] =  0
98                      [195.69, 174.30, 210.56, 3.67, 15.069, 290.0, 337.16, 407.85],      # a = 1 ,
    task_type[l] =  1
99                      [207.91, 192.07, 207.91, 7.23, 0.65, 299.35, 332.17, 394.58],       # a = 2 ,
    task_type[l] =  2
100                     [60.96 , 2.86  , 69.71, 174.30, 188.72, 124.55, 151.76, 305.40], # a = 3 ,
    task_type[l] =  3
101                     [20.47 , 61.72 , 0.65, 210.76, 207.91, 81.51, 115.91, 312.48],      # a = 4 ,
    task_type[l] =  4
102                     [122.14, 151.76, 115.91, 374.65, 332.17, 65.15, 3.69, 4.98],        # a = 5 ,
    task_type[l] =  5
103                     [162.67, 350.36 , 311.02, 398.74, 403.58, 120.38, 50.01, 9.53],  # a = 6 ,
    task_type[l] =  6
104                     [89.63 , 124.55, 81.51, 290.0, 299.35, 3.15, 63.56, 125.41]      # a = 7 ,
    task_type[l] =  7
105                     ])
106
107     # time it takes the AMR to arrive to the loading point, not considering
108     # which parking spot the AMR is at (they are separated by 1 or 2 meters max)
109
110     # task_type[j] =
111     #      0     1       2       3      4       5      6       7      8       9
112     t_d = [0, 241.09, 177.34, 240.36, 89.31, 89.63, 42.58, 129.12]
113     # time it takes the AMR to arrive to the unloading point.
114
115     p = p / 100
116
117     def custom_probability():
118         return np.random.choice([0, 1], p=[(1 - p), p])  # p for 1, 1-p for 0
119
120     # Failure to loading point f_c
121     f_c = [[round((custom_probability() * ss.expon.rvs(1, 1.5) * 60), ndigits=2) for j in set_j] for i
    in set_i]
122     # Failure to unloading point f_d
123     f_d = [[round((custom_probability() * ss.expon.rvs(1, 1.5) * 60), ndigits=2) for j in set_j] for i
    in set_i]
124
125     print('f_c =', f_c)
126     print('f_d =', f_d, '\n')
127
128     Q = 1000 * math.ceil((1 + (max(deadline) - min(time)) / 1000))
129
130     # ------------------------------------------------- MODEL VARIABLES
     -----------------------------------------------
131
132     x = {(i, j): opt_model.addVar(vtype=grb.GRB.BINARY, name="x_{0}_{1}".format(i, j), lb=0)
133          for i in set_i for j in set_j}
134     # Dictionary with indexes i and j, where x_(i)_(j) is a binary variable:
135     # if x_(i)_(j) = 1, AMR i is scheduled to do task j
136     # if x_(i)_(j) = 0, AMR i is not scheduled to do task j
137
138     y = {(i, l, j): opt_model.addVar(vtype=grb.GRB.BINARY, name="y_{0}_{1}_{2}".format(i, l, j), lb=0)
139          for i in set_i for l in set_l for j in set_j}
140     # Dictionary with indexes i, l and j, where y_(i)_(l)_(j) is a binary variable:
141     # if y_(i)_(l)_(j) = 1, task l that precedes task k if both are scheduled for AMR i
142     # if y_(i)_(l)_(j) = 0, task l does not that precedes task k
143
144     p = {(i, l, j): opt_model.addVar(vtype=grb.GRB.CONTINUOUS, name="p_{0}_{1}_{2}".format(i, l, j), lb
    =0)
145          for i in set_i for l in set_l for j in set_j if (l != j)}
146     # Dictionary with index i, l and j where p_(i)_(l)_(j) is a continuous variable
147     # regarding the task duration or processing time, when scheduled for AMR i and
```

Figura D.3: *Script Python* para a otimização do modelo MILP proposto (Parte 3).

File: Robust_Schedule_1.py

```
148     # having the last task l in that AMR
149
150     # Auxiliary restriction to build full_p_3d array
151     full_p = {(i, l, j): opt_model.addVar(vtype=grb.GRB.CONTINUOUS, name="full_p_{0}_{1}_{2}".format(i
    , l, j), lb=0)
152           for i in set_i for l in set_l for j in set_j if (l != j)}
153     # Dictionary with index i, l and j where p_(i)_(l)_(j) is a continuous variable
154     # regarding the task duration or processing time, when scheduled for AMR i and
155     # having the last task l in that AMR
156
157     C = {(i, l, j): opt_model.addVar(vtype=grb.GRB.CONTINUOUS, name="C_{0}_{1}_{2}".format(i, l, j), lb
    =0)
158           for i in set_i for l in set_l for j in set_j if (l != j)}
159     # Dictionary with index i, l and j, where C_(i)_(l)_(j) is a continuous variable and describes
160     # the instant where the AMR arrives to the loading point of an empty rack or
161     # the unloading point of a full rack
162
163     S = {(i, l, j): opt_model.addVar(vtype=grb.GRB.CONTINUOUS, name="S_{0}_{1}_{2}".format(i, l, j), lb
    =0)
164           for i in set_i for l in set_l for j in set_j if (l != j)}
165     # Dictionary with index i, l and j, where S_(i)_(l)_(j) is a continuous variable and
166     # describes the task start time when scheduled for AMR i
167
168     F = {(i, l, j): opt_model.addVar(vtype=grb.GRB.CONTINUOUS, name="F_{0}_{1}_{2}".format(i, l, j), lb
    =0)
169           for i in set_i for l in set_l for j in set_j if (l != j)}
170     # Dictionary with index i, l and j, where F_(i)_(l)_(j) is a continuous variable and
171     # describes the task finishing time, when scheduled for AMR i
172
173     # ------------------------------------------------ MODEL CONSTRAINTS
     ----------------------------------------------
174
175     one_amr = {(j): opt_model.addLConstr(
176         lhs=grb.quicksum(x[i, j] for i in set_i),
177         sense=grb.GRB.EQUAL,
178         rhs=1,
179         name="one_amr_{0}".format(j))
180         for j in set_j
181     }
182
183     amr_tasks = {(i): opt_model.addLConstr(
184         lhs=grb.quicksum(x[i, j] for j in set_j),
185         sense=grb.GRB.GREATER_EQUAL,
186         rhs=1,
187         name="amr_tasks_{0}".format(i))
188         for i in set_i if (n >= m)
189     }
190
191     first_task1 = {(i): opt_model.addLConstr(
192         lhs=grb.quicksum(y[i, -1, j] for j in set_j),
193         sense=grb.GRB.EQUAL,
194         rhs=1,
195         name="first_task1_{0}".format(i))
196         for i in set_i if (n >= m)
197     }  # One AMR can only have 1 first task
198
199     first_task2 = {(i): opt_model.addLConstr(
200         lhs=grb.quicksum(y[i, -1, j] for j in set_j),
201         sense=grb.GRB.LESS_EQUAL,
202         rhs=1,
203         name="first_task2_{0}".format(i))
204         for i in set_i if (n < m)
205     }  # One AMR can only have 1 first task
206
207     first_task3 = {(j): opt_model.addLConstr(
208         lhs=grb.quicksum(y[i, -1, j] for i in set_i),
209         sense=grb.GRB.LESS_EQUAL,
210         rhs=1,
211         name="first_task3_{0}".format(j))
212         for j in set_j if (n > m)
213     }  # One task can only be a first task to one AMR
```

Figura D.4: *Script Python* para a otimização do modelo MILP proposto (Parte 4).

File: Robust_Schedule_1.py

```
214
215     first_task4 = {(j): opt_model.addLConstr(
216         lhs=grb.quicksum(y[i, -1, j] for i in set_i),
217         sense=grb.GRB.EQUAL,
218         rhs=1,
219         name="first_task3_{0}".format(j))
220         for j in set_j if (n <= m)
221     }  # One task can only be a first task to one AMR
222
223
    # ------------------------------------------------------------------------------------------------------------
    ----------
224
225     processing_time_1 = {(i, l, j): opt_model.addLConstr(
226         lhs=p[i, l, j],
227         sense=grb.GRB.EQUAL,
228         rhs=y[i, l, j] * (t_c[task_type[l]][task_type[j]] + load[i][j] + t_d[task_type[j]] + unload[i][
    j]
229                           + f_c[i][j] + f_d[i][j]),
230         name="proctime1_{0}_{1}_{2}".format(i, l, j))
231         for i in set_i for j in set_j for l in set_l if (l != -1) and (j != l)
232     }
233
234     processing_time_2 = {(i, -1, j): opt_model.addLConstr(
235         lhs=p[i, -1, j],
236         sense=grb.GRB.EQUAL,
237         rhs=y[i, -1, j] * (t_c[last_tasktype[i]][task_type[j]] + load[i][j] + t_d[task_type[j]] +
    unload[i][j]
238                            + f_c[i][j] + f_d[i][j]),
239         name="proctime2_{0}_{1}_{2}".format(i, -1, j))
240         for i in set_i for j in set_j
241     }  # Processing time for the first tasks of each AMR, considering the type of
242     # the last task (last_tasktype[i]), in the last schedule
243
244     processing_time_1_full = {(i, l, j): opt_model.addLConstr(
245         lhs=full_p[i, l, j],
246         sense=grb.GRB.EQUAL,
247         rhs=(t_c[task_type[l]][task_type[j]] + load[i][j] + t_d[task_type[j]] + unload[i][j]
248                           + f_c[i][j] + f_d[i][j]),
249         name="proctime1_full_{0}_{1}_{2}".format(i, l, j))
250         for i in set_i for j in set_j for l in set_l if (l != -1) and (j != l)
251     }
252
253     processing_time_2_full = {(i, -1, j): opt_model.addLConstr(
254         lhs=full_p[i, -1, j],
255         sense=grb.GRB.EQUAL,
256         rhs=(t_c[last_tasktype[i]][task_type[j]] + load[i][j] + t_d[task_type[j]] + unload[i][j]
257                           + f_c[i][j] + f_d[i][j]),
258         name="proctime2_full_{0}_{1}_{2}".format(i, -1, j))
259         for i in set_i for j in set_j
260     }
261
262     start_time1 = {(i, l, j): opt_model.addLConstr(
263         lhs=S[i, l, j],
264         sense=grb.GRB.GREATER_EQUAL,
265         rhs=y[i, l, j] * time[i],
266         name="start1_{0}_{1}_{2}".format(i, l, j))
267         for i in set_i for j in set_j for l in set_l if (j != l)
268     }  # In a new schedule, every task begins after the last finish time of every
269     # AMR i (time[i]), in the last schedule
270
271     completion_1 = {(i, l, j): opt_model.addLConstr(
272         lhs=C[i, l, j] - S[i, l, j],
273         sense=grb.GRB.EQUAL,
274         rhs=p[i, l, j] - y[i, l, j] * (delay[j] * (load[i][j] + t_d[task_type[j]] + unload[i][j] + f_d[
    i][j])),
275         name="comp1_{0}_{1}_{2}".format(i, l, j))
276         for i in set_i for j in set_j for l in set_l if (j != l)
277     }
278
279     finish_time2 = {(i, l, j): opt_model.addLConstr(
```

Figura D.5: *Script Python* para a otimização do modelo MILP proposto (Parte 5).

File: Robust_Schedule_1.py

```
280         lhs=F[i, l, j] - S[i, l, j],
281         sense=grb.GRB.GREATER_EQUAL,
282         rhs=0,
283         name="finish2_{0}_{1}_{2}".format(i, l, j))
284         for i in set_i for j in set_j for l in set_l if (l != j)
285     }
286 
287     dline1 = {(i, l, j): opt_model.addLConstr(
288         lhs=C[i, l, j] - deadline[j],
289         sense=grb.GRB.LESS_EQUAL,
290         rhs=0,
291         name="dline1_{0}_{1}_{2}".format(i, l, j))
292         for i in set_i for j in set_j for l in set_l if (l != j)
293     }
294 
295     finish_time1 = {(i, l, j): opt_model.addLConstr(
296         lhs=F[i, l, j],
297         sense=grb.GRB.EQUAL,
298         rhs=y[i, l, j] * (deadline[j] + delay[j] * (load[i][j] + t_d[task_type[j]] + unload[i][j] + f_d
    [i][j])),
299         name="finish1_{0}_{1}_{2}".format(i, l, j))
300         for i in set_i for j in set_j for l in set_l if (j != l)
301     }  # the finish time of every task is equal ou greater than the deadline, since the deadline can't
    be changed
302 
303     # (j != k) and (j != l) and (k != l) so that the precedence is only established between different
    tasks
304 
305 
    # ----------------------------------------------------------------------------------------------------
    ----------
306 
307     precedence_1 = {(i, l, j, k): opt_model.addLConstr(
308         lhs=(S[i, j, k] - S[i, l, j]) + Q * (4 - x[i, j] - x[i, k] - y[i, l, j] - y[i, j, k]),
309         sense=grb.GRB.GREATER_EQUAL,
310         rhs=p[i, l, j],
311         name='prec1_{0}_{1}_{2}_{3}'.format(i, l, j, k))
312         for i in set_i for j in set_j for l in set_l for k in set_k
313         if (n > m) and (j != k) and (j != l) and (k != l)
314     }
315 
316     precedence_2 = {(i, l, j, k): opt_model.addLConstr(
317         lhs=(S[i, l, j] - S[i, j, k]) + Q * (3 - x[i, j] - x[i, k] - y[i, l, j] + y[i, j, k]),
318         sense=grb.GRB.GREATER_EQUAL,
319         rhs=p[i, j, k],
320         name="prec2_{0}_{1}_{2}_{3}".format(i, l, j, k))
321         for i in set_i for j in set_j for l in set_l for k in set_k
322         if (n > m) and (j != k) and (j != l) and (k != l)
323     }
324 
325     precedence_3 = {(i, l, j, k): opt_model.addLConstr(
326         lhs=(F[i, j, k] - F[i, l, j]) + Q * (4 - x[i, j] - x[i, k] - y[i, l, j] - y[i, j, k]),
327         sense=grb.GRB.GREATER_EQUAL,
328         rhs=p[i, j, k],
329         name="prec3_{0}_{1}_{2}_{3}".format(i, l, j, k))
330         for i in set_i for j in set_j for l in set_l for k in set_k
331         if (n > m) and (j != k) and (j != l) and (k != l)
332     }
333 
334     precedence_4 = {(i, l, j, k): opt_model.addLConstr(
335         lhs=(F[i, l, j] - F[i, j, k]) + Q * (3 - x[i, j] - x[i, k] - y[i, l, j] + y[i, j, k]),
336         sense=grb.GRB.GREATER_EQUAL,
337         rhs=p[i, l, j],
338         name="prec4_{0}_{1}_{2}_{3}".format(i, l, j, k))
339         for i in set_i for j in set_j for l in set_l for k in set_k
340         if (n > m) and (j != k) and (j != l) and (k != l)
341     }
342 
343 
    # ----------------------------------------------------------------------------------------------------
    ----------
```

Figura D.6: *Script Python* para a otimização do modelo MILP proposto (Parte 6).

File: Robust_Schedule_1.py

```
344
345     full_p_positive = {(i, l, j): opt_model.addLConstr(
346         lhs=full_p[i, l, j],
347         sense=grb.GRB.GREATER_EQUAL,
348         rhs=0,
349         name="full_p_positive_{0}_{1}_{2}".format(i, l, j))
350         for i in set_i for j in set_j for l in set_l if (l != j)
351     }  # Auxiliary restriction to build full_p_3d array
352
353     p_positive = {(i, l, j): opt_model.addLConstr(
354         lhs=p[i, l, j],
355         sense=grb.GRB.GREATER_EQUAL,
356         rhs=0,
357         name="p_positive_{0}_{1}_{2}".format(i, l, j))
358         for i in set_i for j in set_j for l in set_l if (l != j)
359     }
360
361     S_positive = {(i, l, j): opt_model.addLConstr(
362         lhs=S[i, l, j],
363         sense=grb.GRB.GREATER_EQUAL,
364         rhs=0,
365         name="S_positive_{0}_{1}_{2}".format(i, l, j))
366         for i in set_i for j in set_j for l in set_l if (l != j)
367     }
368
369     C_positive = {(i, l, j): opt_model.addLConstr(
370         lhs=C[i, l, j],
371         sense=grb.GRB.GREATER_EQUAL,
372         rhs=0,
373         name="C_positive_{0}_{1}_{2}".format(i, l, j))
374         for i in set_i for j in set_j for l in set_l if (l != j)
375     }
376
377     F_positive = {(i, l, j): opt_model.addLConstr(
378         lhs=F[i, l, j],
379         sense=grb.GRB.GREATER_EQUAL,
380         rhs=0,
381         name="F_positive_{0}_{1}_{2}".format(i, l, j))
382         for i in set_i for j in set_j for l in set_l if (l != j)
383     }
384
385     # ----------------------------------------- MODEL OBJECTIVE FUNCTION
    -----------------------------------------
386
387     # Objective function to minimize - Task Duration
388     objective = grb.quicksum((F[i, l, j] - S[i, l, j])
389                              for i in set_i
390                              for l in set_l
391                              for j in set_j
392                              if (l != j))
393
394     opt_model.ModelSense = grb.GRB.MINIMIZE
395
396     opt_model.setObjective(objective)
397
398     # model presolve
399     reduced_model = opt_model.presolve()
400     reduced_model.optimize()
401
402     print('\nPresolve Status =', reduced_model.Status)
403
404     if reduced_model.Status != 2:
405
406         if reduced_model.Status == 3:
407             print('\n-------------- SOLUTION NOT FOUND (INFEASIBLE) --------------')
408             reduced_model.computeIIS()
409
410             reduced_model.write(f'{str_current_datetime}_infeasible_model_num{number}.ilp')
411
412             print('\nThe following constraints and variables are in the IIS:')
413             for c in reduced_model.getConstrs():
```

Figura D.7: *Script Python* para a otimização do modelo MILP proposto (Parte 7).

File: Robust_Schedule_1.py

```
414                 if c.IISConstr: print(f'{c.constrname}: {reduced_model.getRow(c)} {c.Sense} {c.RHS}\n'
    )
415
416             for v in reduced_model.getVars():
417                 if v.IISLB: print(f'LB - {v.varname} ≥ {v.LB}\n')
418                 if v.IISUB: print(f'UB - {v.varname} ≤ {v.UB}\n')
419             schedule = []
420
421         elif reduced_model.Status == 4:
422             print('\n-------------- MODEL IS INFEASIBLE OR UNBOUNDED --------------')
423             for c in reduced_model.getConstrs():
424                 print(f'{c.ConstrName}: {reduced_model.getRow(c)} {c.Sense} {c.RHS}\n')
425             schedule = []
426
427         elif reduced_model.Status == 5:
428             print('\n-------------- MODEL IS UNBOUNDED --------------')
429             for c in reduced_model.getConstrs():
430                 print(f'{c.ConstrName}: {reduced_model.getRow(c)} {c.Sense} {c.RHS}\n')
431             print(reduced_model.InfUnbdInfo)
432             schedule = []
433
434     else:
435         print('\n-------------- SOLUTION FOUND --------------')
436         reduced_model.write(f'{str_current_datetime}_presolve_num{number}.lp')
437         reduced_model.write(f'{str_current_datetime}_presolve_num{number}.mps')
438
439
440     # model optimization
441     opt_model.optimize()
442
443     # To save and export model to be read by other solvers
444     opt_model.write(f"{str_current_datetime}_mymodel_num{number}.mps")
445
446     opt_model.printStats()
447
448     print('\nStatus =', opt_model.Status)
449
450     if opt_model.Status != 2:
451
452         if opt_model.Status == 3:
453             print('\n-------------- SOLUTION NOT FOUND (INFEASIBLE) --------------')
454             opt_model.computeIIS()
455
456             opt_model.write(f'{str_current_datetime}_infeasible_model_num{number}.ilp')
457
458             print('\nThe following constraints and variables are in the IIS:')
459             for c in opt_model.getConstrs():
460                 if c.IISConstr: print(f'{c.constrname}: {opt_model.getRow(c)} {c.Sense} {c.RHS}\n')
461
462             for v in opt_model.getVars():
463                 if v.IISLB: print(f'LB - {v.varname} ≥ {v.LB}\n')
464                 if v.IISUB: print(f'UB - {v.varname} ≤ {v.UB}\n')
465             schedule = []
466
467         elif opt_model.Status == 4:
468             print('\n-------------- MODEL IS INFEASIBLE OR UNBOUNDED --------------')
469             for c in opt_model.getConstrs():
470                 print(f'{c.ConstrName}: {opt_model.getRow(c)} {c.Sense} {c.RHS}\n')
471             schedule = []
472
473         elif opt_model.Status == 5:
474             print('\n-------------- MODEL IS UNBOUNDED --------------')
475             for c in opt_model.getConstrs():
476                 print(f'{c.ConstrName}: {opt_model.getRow(c)} {c.Sense} {c.RHS}\n')
477             print(opt_model.InfUnbdInfo)
478             schedule = []
479
480     else:
481         print('\n-------------- SOLUTION FOUND --------------')
482
483         # Write last solution to file
```

Figura D.8: *Script Python* para a otimização do modelo MILP proposto (Parte 8).

File: Robust_Schedule_1.py

```
484         opt_model.write(f'{str_current_datetime}_opt_solution_num{number}.sol')
485 
486         # code to check each variable value after optimization
487         # for v in opt_model.getVars():
488         #     print(f"{v.VarName} = {round(v.X, 2)}")
489         # print('\n')
490         for v in opt_model.getConstrs():
491             print(f"{v.ConstrName} = {v.Slack}")
492             print(f'{v.ConstrName}: {opt_model.getRow(v)} {v.Sense} {v.RHS}\n')
493 
494 
495         # IMPORTANT: Inside each list and array, the order for the "for" loops is the following:
496         # for i in set_i
497         # for l in set_l
498         # for j in set_j
499         # if y_3d[i, l, j] == 1 and amr_list[i][j] == 1
500 
501         x_list = [v.X for v in x.values()]
502         print("x_list = ", x_list, '\n')
503 
504         amr_list = [x_list[x:x + n]
505                     for x in range(0, len(x_list), n)]
506         print('amr_list\n', amr_list, "\n")
507 
508         y_3d = np.zeros((m, n + 1, n))
509         for (a, b, c), value in y.items():
510             y_3d[a, b, c] = value.X
511         # Note: -1 is the "last b" instead of the first
512         print("y_3d\n", y_3d, '\n')
513 
514         amr_final = [amr[i]
515                      for i in set_i
516                      for l in set_l
517                      for j in set_j
518                      if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
519         print('amr_final = ', amr_final, "\n")
520 
521         load_ij_final = [round(load[i][j], ndigits=3)
522                          for i in set_i
523                          for l in set_l
524                          for j in set_j
525                          if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
526 
527         t_d_ij_final = [t_d[task_type[j]]
528                         for i in set_i
529                         for l in set_l
530                         for j in set_j
531                         if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
532 
533         unload_ij_final = [round(unload[i][j], ndigits=3)
534                            for i in set_i
535                            for l in set_l
536                            for j in set_j
537                            if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
538 
539         f_c_final = [round(f_c[i][j], ndigits=2)
540                      for i in set_i
541                      for l in set_l
542                      for j in set_j
543                      if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
544 
545         f_d_final = [round(f_d[i][j], ndigits=2)
546                      for i in set_i
547                      for l in set_l
548                      for j in set_j
549                      if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
550 
551         p_3d = np.zeros((m, n + 1, n))
552         for (a, b, c), value in p.items():
553             p_3d[a, b, c] = round(value.X, ndigits=2)
554         print('p_3d\n', p_3d, '\n')
```

Figura D.9: *Script Python* para a otimização do modelo MILP proposto (Parte 9).

File: Robust_Schedule_1.py

```
555         p_final = [p_3d[i, l, j]
556                    for i in set_i
557                    for l in set_l
558                    for j in set_j
559                    if y_3d[i, l, j] == 1 and amr_list[i][j] == 1]
560 
561         full_p_3d = np.zeros((m, n + 1, n))
562         for (a, b, c), value in full_p.items():
563             full_p_3d[a, b, c] = round(value.X, ndigits=2)
564         print('full_p_3d\n', full_p_3d, '\n')
565 
566         if len(p_final) == len(task_list[0]):
567 
568             # print('amr_final = ', amr_final, "\n")
569 
570             # print("p_final = ", p_final, '\n')
571             #
572             # print("load_ij_final = ", load_ij_final, "\n")
573             #
574             # print("t_d_ij_final = ", t_d_ij_final, "\n")
575             #
576             # print("unload_ij_final = ", unload_ij_final, "\n")
577             #
578             # print("f_c_final = ", f_c_final, '\n')
579             #
580             # print("f_d_final = ", f_d_final, '\n')
581 
582             diff = [round(
583                 p_final[j] - load_ij_final[j] - t_d_ij_final[j] - unload_ij_final[j] - f_c_final[j] -
    f_d_final[j],
584                 ndigits=2)
585                     for j in set_j]
586             print("sum = ", diff, '\n')
587 
588             S_3d = np.zeros((m, n + 1, n))
589             for (a, b, c), value in S.items():
590                 S_3d[a, b, c] = round(value.X, ndigits=2)
591             print("S_3d\n", S_3d, '\n')
592             S_final = [S_3d[i, l, j]
593                        for i in set_i
594                        for l in set_l
595                        for j in set_j
596                        if z_3d[i, l, j] == 1]
597             print('S_final', S_final, "\n")
598 
599             F_3d = np.zeros((m, n + 1, n))
600             for (a, b, c), value in F.items():
601                 F_3d[a, b, c] = round(value.X, ndigits=2)
602             print("F_3d\n", F_3d, '\n')
603             F_final = [F_3d[i, l, j]
604                        for i in set_i
605                        for l in set_l
606                        for j in set_j
607                        if z_3d[i, l, j] == 1]
608             print('F_final', F_final, "\n")
609 
610             C_3d = np.zeros((m, n + 1, n))
611             for (a, b, c), value in C.items():
612                 C_3d[a, b, c] = round(value.X, ndigits=2)
613             print("C_3d\n", C_3d, '\n')
614             C_final = [C_3d[i, l, j]
615                        for i in set_i
616                        for l in set_l
617                        for j in set_j
618                        if z_3d[i, l, j] == 1]
619             print('C_final', C_final, "\n")
620 
621             task_ID_final = [task_ID[j]
622                              for i in set_i
623                              for l in set_l
624                              for j in set_j
```

Figura D.10: *Script Python* para a otimização do modelo MILP proposto (Parte 10).

File: Robust_Schedule_1.py

```
625                                 if z_3d[i, l, j] == 1]
626             print('task_ID_final', task_ID_final, "\n")
627 
628             task_type_final = [task_type[j]
629                                for i in set_i
630                                for l in set_l
631                                for j in set_j
632                                if z_3d[i, l, j] == 1]
633             print('task_type_final', task_type_final, "\n")
634 
635             deadline_final = [deadline[j]
636                               for i in set_i
637                               for l in set_l
638                               for j in set_j
639                               if z_3d[i, l, j] == 1]
640             print('deadline_final', deadline_final, "\n")
641 
642             schedule_print = [amr_final, task_ID_final, task_type_final, p_final, diff, load_ij_final
643                 , t_d_ij_final, unload_ij_final, f_c_final, f_d_final, S_final, C_final, deadline_final
    , F_final]
644 
645             col = ['AMR', 'Task ID', 'Type', 'Duration', 't_c', 'Load'
646                 , 't_d', 'Unload', 'f_c', 'f_d', 'Start', 'Completion', 'Deadline', 'Finish']
647 
648             # Show Schedule as a Dataframe
649             df = pd.DataFrame(np.array(schedule_print).T, columns=col)
650             pd.set_option('display.max_columns', None)
651 
652             df = pd.DataFrame.sort_values(df, ['AMR'], ascending=True, ignore_index=True)
653 
654             print(df)
655 
656             print('Time = ', round(clock.time() - start_clock, 2), '\n')
657 
658             end_clock = [round(clock.time() - start_clock, 2) for j in set_j]
659             Obj_Solution = [round(opt_model.objVal, 2) for j in set_j]
660             Q_final = [Q for j in set_j]
661 
662             schedule = [amr_final, task_ID_final, task_type_final, S_final, F_final, deadline_final
663                 , f_c_final, f_d_final, end_clock, Obj_Solution, Q_final, C_final]
664 
665         else:
666             print('ERROR: X != Y')
667             schedule = []
668 
669         print('\nObjective Value = ', round(opt_model.objVal, ndigits=2))
670 
671         print('Solution Count = ', opt_model.SolCount)
672         print('\nQ = ', Q, '\n')
673 
674 
675     return schedule
676 
```

Figura D.11: *Script Python* para a otimização do modelo MILP proposto (Parte 11).

File: Test.py

```python
from Robust_Schedule_1 import robust_schedule
from Plot_Schedule import plot_schedule
import numpy as np
import sys
from datetime import datetime

# Test file with various tasks to schedule(varies the number of tasks to schedule)
# These task lists where produced by a Flexsim process flow

# get current date and time
current_datetime = datetime.now().strftime("%Y-%m-%d %H-%M-%S")

# convert datetime obj to string
str_current_datetime = str(current_datetime)

stdoutOrigin = sys.stdout
sys.stdout = open(f"{str_current_datetime}.txt", "w")

# Test file with various tasks to schedule(varies the number of tasks to schedule)
# These task lists where produced by Flexsim

# Available AMR's (uncomment the array to test)

# amr = ["AMR", "AMR_2", "AMR_3"]
# amr = ["AMR", "AMR_2"]
# amr = ["AMR"]

p = 1  # 1% probability of failure

time = [100, 100, 100, 100]

last_tasktype = [0, 0, 0, 0]

# Uncomment/Modify the task_list to test

# 3 tasks
# task_list = np.array([
#     [1,2,3],  # task_id
#     [6,5,7],  # task_type
#     [0,0,0],  # delay
#     [900,850,1050],  # deadline
# ])

# 2 tasks
# task_list = np.array([
#     [1, 2],  # task_id
#     [6, 2],  # task_type
#     [0, 0],  # delay
#     [900, 900],  # deadline
# ])

# 1 task
# task_list = np.array([
#     [1],  # task_id
#     [6],  # task_type
#     [0],  # delay
#     [900],  # deadline
# ])

# number of instances
int_num = 100

for i in range(int_num):

    print(f'\n================= ITERATION {i+1} =================\n')

    number = i + 1

    # Generates de schedule
    schedule = robust_schedule(amr, task_list, time, last_tasktype, p, number)

    # Schedule Plot
    plot_schedule(schedule, number)

sys.stdout.close()
sys.stdout = stdoutOrigin

exit()
```

Figura D.12: *Script Python* de teste do modelo MILP proposto.

File: Plot_Schedule.py

```python
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime


def plot_schedule(task_list, number):

    if len(task_list) == 0:
        return 0

    else:
        # Extract the first sublist and remove duplicates to create the first list
        amr = list(set(task_list[0]))
        amr.sort()

        # Initialize empty lists for second, third and fourth lists
        task_id = [[] for _ in range(len(amr))]
        start_time = [[] for _ in range(len(amr))]
        end_time = [[] for _ in range(len(amr))]
        deadline = [[] for _ in range(len(amr))]
        end_clock = [[] for _ in range(len(amr))]


        # Iterate over the original list and populate all lists
        for i in range(len(task_list[0])):
            index = amr.index(task_list[0][i])
            task_id[index].append(task_list[1][i])
            start_time[index].append(task_list[3][i])
            end_time[index].append(task_list[4][i])
            deadline[index].append(task_list[5][i])
            end_clock[index] = task_list[8][i]

        min_start = min(start_time)[0]
        # max_end = max(end_time)[0]

        time = end_clock[0]

        # Define the labels for the tasks
        task_labels = task_id

        # Plot the schedule
        fig, ax = plt.subplots(figsize=(10, 5))
        for i in range(len(start_time)):
            for j in range(len(start_time[i])):
                start = start_time[i][j]
                end = end_time[i][j]
                dd = deadline[i][j]
                dur = end - start
                task_label = f'{task_labels[i][j]}: {round(dur, 2)} s'
                ax.barh(i, end - start, left=start, height=0.25,
                        color='gray', align='center', alpha=0.8,
                        edgecolor='black', linewidth=1.30)
                ax.text(start + (end - start) / 2, i, task_label, ha='center',
                        va='center', color='black', fontsize=10)

                # Marking the deadline on the chart, if needed
                ax.axvline(x=dd, color='r', linestyle='dashed')
                ax.text(x=dd + 5, y=(len(amr) - 0.5 - 0.01), s=f'{dd} s', color='r')

                from matplotlib.lines import Line2D
                colors = ['red']
                lines = [Line2D([0], [0], color=c, linewidth=1.5, linestyle='dashed') for c in colors]
                labels = ['Task Deadline']
                plt.legend(lines, labels)


        ax.set_xlabel('Time (seconds)')
        ax.set_ylabel('Robot')
        ax.set_yticks([x for x in range(len(amr))])
        ax.set_yticklabels(amr, fontdict={'fontsize': 10,
                                          'verticalalignment': 'center'})
```

Figura D.13: *Script Python* para gerar o *plot* da solução ótima (Parte 1)

File: Plot_Schedule.py

```
72          ax.set_ylim(-0.5, len(amr) - 0.5)
73          ax.invert_yaxis()
74
75          ax.set_xlim(left=(min_start - 100))
76
77          ax.set_title(f'AMR Schedule N°{number} (Solving time = {time}s)', fontdict={'fontsize': 12,
78                                                                                      'fontweight': 'bold',
79                                                                                      'verticalalignment': 'center'})
80
81          # get current date and time
82          current_datetime = datetime.now().strftime("%Y-%m-%d %H-%M-%S")
83
84          # convert datetime obj to string
85          str_current_datetime = str(current_datetime)
86
87          plt.savefig(f'{str_current_datetime}_schedule_{number}.jpg', dpi=1000, bbox_inches='tight')
88
89          # plt.show()
90
91          plt.close()
92
93      return 0
```

Figura D.14: *Script Python* para gerar o *plot* da solução ótima (Parte 2)

File: load_unload_sample.py

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns


def load_unload_sample(sample_size):
    path = r'final_sample_load_unload_2.csv'
    df = pd.read_csv(path)
    d = np.array(df)

    # Transform it into a 1D array
    sample = d.flatten()

    # Create an empirical distribution (PMF) from the sample
    unique_values, counts = np.unique(sample, return_counts=True)
    empirical_distribution = counts / len(sample)

    # Sample from the empirical distribution
    random_sample = np.random.choice(unique_values, size=sample_size, p=empirical_distribution)

    return random_sample
```

Figura D.15: *Script Python* para gerar a Distribuição Empírica que modela os dados de *load/unload*. A função "load_unload_sample"é depois importada no *script python* principal (Figura D.1).

APÊNDICE E

Desenvolvimento das Iterações

Este capítulo apresenta informações adicionais utilizadas nas iterações do desenvolvimento do modelo de Calendarização Robusta proposto. Cada secção é dedicada a uma iteração específica, fornecendo detalhes sobre as diferentes técnicas e abordagens exploradas durante o processo de iterativo de desenvolvimento e aplicação do modelo ao Projeto AGiLE. Dada a breve descrição da Iteração 5, esta não tem uma secção dedicada neste capítulo.

E.1 Iteração 1: Problema Original

A alocação das tarefas do conjunto (1) a $m = 1$ robô foi conseguida nesta iteração. Quando são pedidos mais do que um planeamento (iterando o processo de otimização e gerando 100 planeamentos, dada a natureza estocástica de alguns parâmetros), todos os planeamentos criados para o conjunto (1) foram conseguidos.

Ao alocar o mesmo conjunto (1) a $m = 2$ robôs, os resultados mostram que, se se iterar o processo 2 vezes e gerar 2 calendários, nem todas as soluções ótimas geram alocações. Era esperado que a tarefa fosse alocada ao robô com menor duração prevista, o que nem sempre acontece.

Uma tarefa é alocada, para determinados i, l e j, se $x_{ij} = 1$ e $y_{ilj} = 1$. No caso do conjunto (1) e $m = 1$, como esta condição ocorre por imposição das restrições de capacidade, a tarefa é alocada corretamente. No caso do conjunto (1) e $m = 2$ ou do conjunto (1) e $m = 3$, o mesmo já não acontece em todas ao planeamentos. Pode usar-se o problema associado ao conjunto (1) e $m = 2$, mais simples, para perceber o que está a ocorrer (Figura E.1).

Apenas o planeamento 1 tem correspondência entre as variáveis x_{ij} e y_{ilj}, fazendo com que a tarefa seja alocada ao robô correspondente. No planeamento 2, $y_{0-10} = 1$, mas $x_{00} = 0$ e $x_{10} = 1$, o que faz com que a tarefa não tenha sido alocada ao robô. O *array* auxiliar "*full_p_3d*"mostra que alocar a tarefa do planeamento 1 ao robô $i = 1$ e a tarefa do planeamento 2 ao robô $i = 0$ minimiza a função objetivo em ambos os planeamentos, pois são os robôs com menor duração prevista da tarefa. Por isso, seria de esperar que, no planeamento 2, $x_{00} = 1$, o que não acontece.

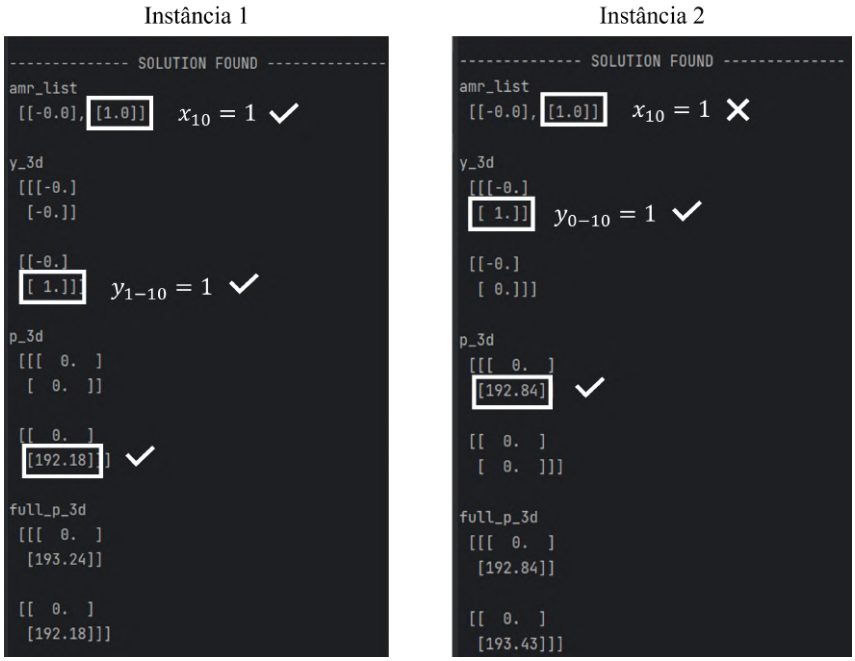


Figura E.1: Resultados de 2 iterações da alocação do conjunto de tarefas (1) a $m = 2$ robôs. O *array* "*full_p_3d*"é um *array* proveniente de uma variável auxiliar, que guarda o valor da variável p_{ilj} sem ser afetado pela variável y_{ilj}, como a Equação 4.11 mostra.

É de notar que nos casos do conjunto (1) com $m = 2$ e $m = 3$, é sempre o robô com o maior índice i para o qual $x_{i0} = 1$, mesmo em planeamentos onde a tarefa é alocada ou não. No *script python* das Figuras D.3, D.4, D.5 e D.6, as variáveis de decisão e restrições utilizam um ciclo *for* para percorrer os dicionários onde estas são guardadas, com os respetivos índices i, l, j e k. Portanto, existe uma ordem pela qual os conjuntos de índices são percorridos. Estes são gerados pela função *range*, que cria uma sequencia bem definida dos elementos do conjunto (Figura D.2). Contudo, ao usar um *range* que cria a sequência $\{m-1, \ldots, 0\}$, o resultado mantém-se o mesmo.

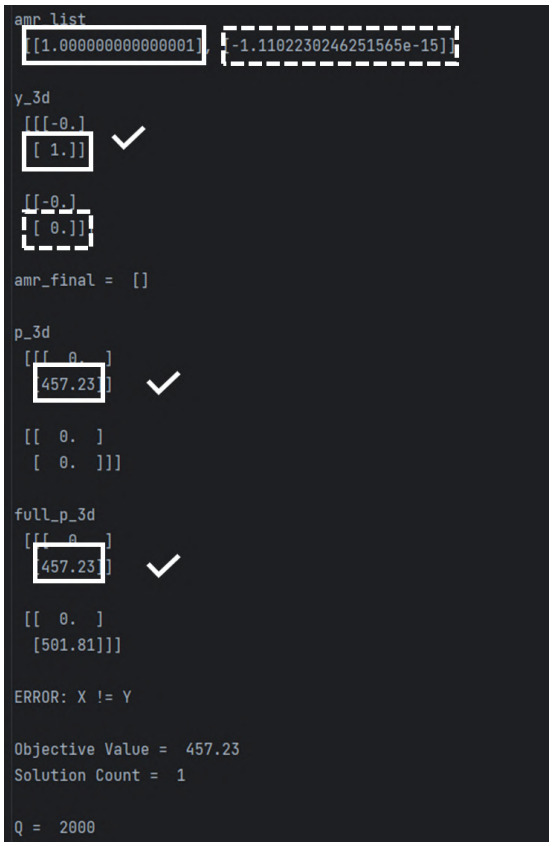
E.2 Iteração 2: Restrições de Sequenciamento que dependem de *n* e *m*

Na Iteração 2, dado que existe um conflito entre as variáveis x_{ij} e y_{ilj} nas Restrições de Sequenciamento, sugere-se a separação dos casos em que $(n \leq m)$ e $(n > m)$ ao adicionar outra restrição a este conjunto.

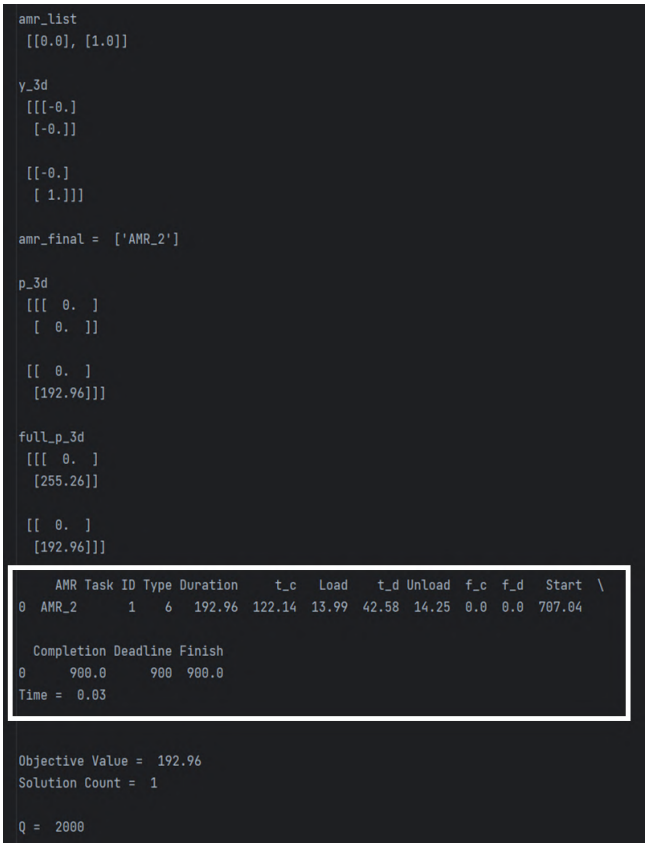
Ao adicionar a Equação (5.3), o conjunto (1) é alocado corretamente a $m = 1$ robôs, tal como acontece antes da adição desta equação. Quando o mesmo conjunto é alocado a $m = 2$ robôs, a integralidade da variável x_{ij} é infringida. A Figura E.2 apresenta dois exemplos de instância s para os casos de falha e sucesso na alocação de uma tarefa.

Para além da Equação (5.3), foram testadas outras variantes desta, sem sucesso, modificando o termo $Q \times (1 - x_{ij} + y_{ilj})$, como $Q \times (2 + x_{ij} + y_{ilj})$. Note-se que retirar a Equação (5.3) e manter a condição $(n > m)$ nas Equações (4.17), (4.18), (4.19) e (4.20) também não é uma solução, pois é necessário que, para $(n \leq m)$, exista também uma relação entre x_{ij} e y_{ilj} numa restrição de sequenciamento.

Em 10 instâncias, apenas uma alocou a tarefa do conjunto (1) com sucesso, imprimindo o planeamento rodeado por um retângulo branco na Figura E.2b (utilizando a estrutura de



(a) Instância onde não houve alocação



(b) Instância onde houve alocação

Figura E.2: Resultado da alocação da tarefa do conjunto (1) a $m = 2$ robôs. Em 10 instâncias, apenas uma foi bem sucedida na alocação correta da tarefa do conjunto (1). O parâmetro $Q = 2000$ é igual em todas as instâncias.

dados *Dataframe*, da biblioteca *Pandas*), para que se possa consultar as características de cada tarefa planeada. Nas restantes instâncias, o *output* é semelhante ao da Figura E.2a, onde o valor final das variáveis x_{ij} não é inteiro, mas é muito próximo do valor inteiro esperado. A tarefa $j = 0$, quando alocada ao robô $i = 0$ após a tarefa $l = -1$ ($y_{0-10} = 1$) está associada à duração prevista que minimiza a função objetivo ($p_{0-10} = 457,23\ s$), portanto espera-se que $x_{00} = 1$, não $x_{00} = 1,000000000000001$. Nestes casos, para além de ser uma quebra da integralidade da variável binária, é perigoso arredonda-la para o número inteiro mais próximo, pois pode tornar o modelo inviável.

Este problema é considerado um problema numérico, que resulta em alocações inconsistentes com os resultados esperados. O Gurobi Optimizer (GRB) elabora este tipo de problemas, agrupando-os em categorias para facilitar a sua exposição e resolução (Gurobi Optimization, 2023a, p. 983). O incumprimento da integralidade da variável x_{ij} está associado à utilização de restrições do tipo *Big M*, que tem as suas limitações e desvantagens. Nomeadamente, como $x_{00} = 1$ torna o modelo inviável e $x_{00} = 1,000000000000001$ está dentro da tolerância definida pelo parâmetro *Integer Feasibility Tolerance* (*IntFeasTol*), o modelo interpreta $1,000000000000001$ como o número inteiro 1 e apresenta $x_{00} = 1,000000000000001$ como uma solução viável. Esta tolerância possibilita também $y_{0-10} = 1$, fazendo cumprir a Equação (5.3) para os índices $i = 0, l = -1$ e $j = 0$.

O parâmetro *IntFeasTol* permite assumir que o resultado $x_{00} = 1,000000000000001$ é inteiro

e igual a 1, desde que o módulo da diferença entre ambos esteja dentro da tolerância do *solver* (Gurobi Optimization, 2023a, p. 836). Por definição, este parâmetro tem um valor *default* de 10^{-5}, valor mínimo de 10^{-9} e valor máximo de 10^{-1}. Outro parâmetro que influência o cumprimento das restrições de integralidade das variáveis é o parâmetro binário *IntegralityFocus* cujo valor *default* é 0 (Gurobi Optimization, 2023a, p. 835). Quando ativado (*IntegralityFocus*= 1), é pedido ao *solver* para encontrar soluções onde são cumpridas todas a restrições de integralidade e que mantenham o modelo viável, arredondando as variáveis cuja restrição requer que esta sejam inteiras. Uma desvantagem da utilização destes parâmetros é a possibilidade de piorar a performance da otimização. No entanto, mesmo utilizando o valor mínimo do parâmetro *IntFeasTol* e ativando o parâmetro *IntegralityFocus*, o resultado mantém o incumprimento da integralidade da variável x_{ij}.

É imperativo que as variáveis x_{ij} e y_{ilj} sejam binárias, portanto a reformulação destas não é possível. Antes da otimização do modelo, por predefinição, o GRB inicia a rotina de *presolve* de forma a acelerar o processo de otimização e melhorar a performance do *solver*. O *presolve* pode também ser uma fonte dos problemas numéricos observados. Um dos parâmetros relacionados com o mesmo é o parâmetro *Aggregate* (*Aggregate*=1 por definição). Quando ativado, é responsável pela agregação de pares de variáveis simétricas numa única variável (Achterberg et al., 2020, p. 44). Ao desativar este parâmetro (*Aggregate*=0), a alocação do conjunto (1) a $m = 2$ robôs é feita com sucesso para todas as 100 instâncias geradas (Figura E.3).

```
amr_list
 [[1.0], [0.0]]

y_3d
 [[[-0.]
  [ 1.]]

 [[-0.]
  [ 0.]]]

amr_final =  ['AMR']

p_3d
 [[[  0.  ]
  [193.06]]

 [[  0.  ]
  [  0.  ]]]

full_p_3d
 [[[  0.  ]
  [193.06]]

 [[  0.  ]
  [193.39]]]

   AMR Task ID Type Duration     t_c   Load    t_d Unload  f_c  f_d   Start  \
0  AMR       1    6   193.06  122.14  14.58  42.58  13.76  0.0  0.0  706.94

  Completion Deadline Finish
0      900.0      900  900.0
Time =  0.09


Objective Value =  193.06
Solution Count =  2

Q =  2000
```

Figura E.3: Exemplo de uma instância gerada pela alocação da tarefa do conjunto (1) a $m = 2$ robôs (parâmetro *Aggregate*=0).

Analisando os ficheiros *.lp* produzidos após o *presolve* (Figura E.4), na Figura E.4a pode observar-se que as variáveis y_{ilj} e C_{ilj} são as necessárias para resolver problema inicial. Na Figura E.4b, a variável C_{ilj} é substituída por S_{ilj}, o que indica que as variáveis S_{ilj} e F_{ilj} foram agregadas numa única variável e resolvidas durante o processo de *presolve*. S_{ilj} e F_{ilj} passam a ser as únicas variáveis nas restrições do sub-problema.

Uma semelhança entre os dois sub-problemas é a utilização de variáveis não nulas com o índice $l = -1$ (Figura E.4b). As variáveis y_{000} e y_{100}, embora não utilizem o índice $l = -1$, têm um coeficiente nulo e foram excluídas das restrições na secção *Subject To*, portanto não terão qualquer influência na função objetivo durante a otimização. O mesmo acontece na Figura E.4a, onde uma das variáveis com influência no valor da função objetivo e restrições é a variável y_{0-10}, responsável pela alocação da tarefa $j = 0$ e à qual será atribuído o valor 1, no final da otimização.

Em ambos os casos, as variáveis x_{ij} e p_{ilj}, são excluídas do sub-problema para quaisquer índices i, l e j, o que leva a concluir que isto ocorre por outro processo que não a agregação de pares de variáveis e que decorre durante o *presolve*. Estas variáveis estão diretamente relacionadas com as variáveis y_{ilj}, portanto é de esperar que fossem reduzidas do problema.

```
\ Model MILP Model_pre
\ LP format - for model browsing. Use MPS format to capture full model detail.
Minimize
  - 44.58 y_0_-1_0 + 0 y_0_0_0 + 0 y_1_0_0 - C_0_-1_0 - C_1_-1_0
  + 1401.81 Constant
Subject To
 start1_0_-1_0: 557.23 y_0_-1_0 - C_0_-1_0 <= 0
 start1_1_-1_0: - 601.81 y_0_-1_0 - C_1_-1_0 <= -601.81
 finish2_0_-1_0: - 900 y_0_-1_0 + C_0_-1_0 <= 0
 finish2_1_-1_0: 900 y_0_-1_0 + C_1_-1_0 <= 900
Bounds
 C_0_-1_0 <= 900
 C_1_-1_0 <= 900
 Constant = 1
Binaries
 y_0_-1_0 y_0_0_0 y_1_0_0
End
```

(a) *Aggregate*= 1

```
\ Model MILP Model_pre
\ LP format - for model browsing. Use MPS format to capture full model detail.
Minimize
  0 y_0_0_0 + 0 y_1_0_0 - S_0_-1_0 - S_1_-1_0 + 900 Constant
Subject To
 start1_0_-1_0: - S_0_-1_0 + 100 F_0_-1_0 <= 0
 start1_1_-1_0: - S_1_-1_0 - 100 F_0_-1_0 <= -100
 comp1_0_-1_0: C_0_-1_0 - S_0_-1_0 - 193.06 F_0_-1_0 = 0
 comp1_1_-1_0: C_1_-1_0 - S_1_-1_0 + 193.39 F_0_-1_0 = 193.39
 finish2_0_-1_0: S_0_-1_0 - 706.94 F_0_-1_0 <= 0
 finish2_1_-1_0: S_1_-1_0 + 706.61 F_0_-1_0 <= 706.61
Bounds
 C_0_-1_0 <= 900
 C_1_-1_0 <= 900
 S_0_-1_0 <= 706.94
 S_1_-1_0 <= 706.61
 Constant = 1
Binaries
 y_0_0_0 y_1_0_0 F_0_-1_0
End
```

(b) *Aggregate*= 0

Figura E.4: Ficheiro *.lp* do sub-problema do conjunto (1) e $m = 2$ robôs, após o *root presolve*

A desativação do parâmetro *Aggregate* não implica que, para outro conjunto de tarefas, os problemas de integralidade não voltem a ocorrer. Como explicado anteriormente, este tipo de problemas está intimamente ligado á rotina de *presolve* do *solver* e, a cima de tudo, ao facto do problema ser ou não viável quando as variáveis inteiras tomam os valores esperados. O *solver* permite também desligar por completo a rotina de *presolve*, o que traz algumas penalizações de performance e, mesmo assim, não garante a viabilidade do problema (Gurobi Optimization, 2023a). Dependendo dos resultados obtidos, cabe ao utilizador do *solver* decidir se aceita esta instabilidade de soluções e se a ocorrência deste tipo de problemas pode ser ultrapassada de outra forma, como por exemplo, o ajuste de alguns parâmetros do *solver* ou coeficientes do modelo, de forma a torna-lo menos sensível. Dados os resultados promissores da desativação do parâmetro *Aggregate*, este permanecerá desativado nas restantes iterações.

E.3 Iteração 3: Alocação de tarefas após a primeira tarefa de cada robô

No seguimento da Iteração 2, quando é alocado o conjunto (2) a $m = 1$ robôs, apenas a tarefa $j = 0$ é alocada, pois é a tarefa com um *deadline* mais próximo (Figura E.5). Recorrendo novamente aos *arrays* auxiliares para visualizar o valor final das variáveis, como $j = 0$ é a primeira tarefa do robô $i = 0$ (ou seja, $l = -1$) e a variável y_{0-10} é restringida pelas Equações (4.7) e (4.9), o resultado $y_{0-10} = 1$ era esperado.

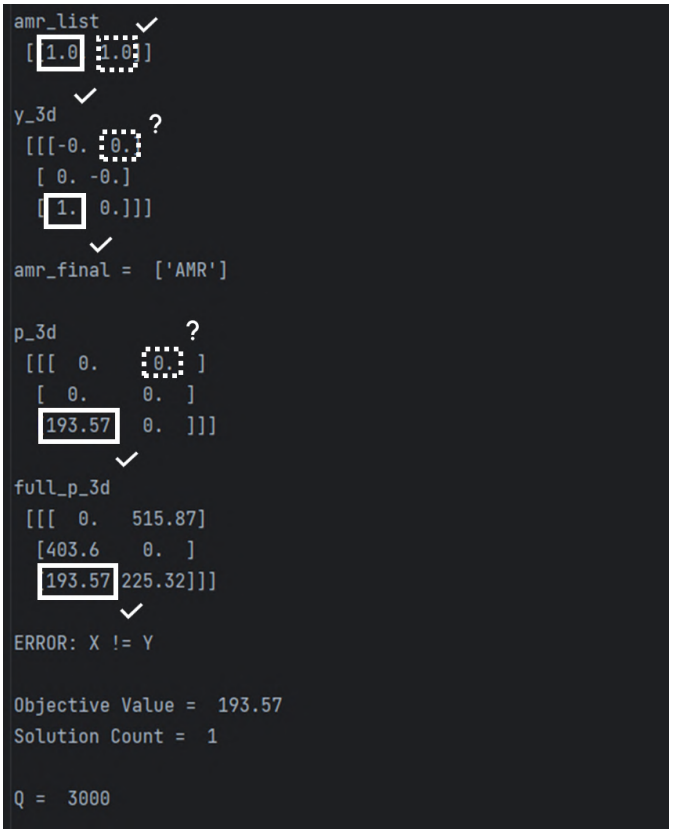


Figura E.5: Resultado da alocação da tarefa do conjunto (2) a $m = 1$ robôs.

No entanto, a tarefa $j = 1$, que se esperava estar alocada ao robô $i = 0$ após a tarefa $j = 0$. (ou seja, $l = 0$) não foi alocada, pois $y_{001} = 0$ e $x_{01} = 1$. Para $j = 1$ estar alocada ao robô: $y_{001} = 1 \wedge x_{01} = 1$. Como só está uma tarefa alocada, o valor ótimo da função objetivo corresponde apenas à duração da tarefa $j = 0$.

Como não é possível a evolução do algoritmo *Branch-and-Cut* durante a otimização, é necessário usar o valor final das variáveis e de ambos os lados das restrições para perceber qual a fonte da incongruência entre variáveis. Ao resolver as Equações (4.17), (4.18), (4.19) e (4.20), sabendo que $x_{00} = x_{01} = y_{0-10} = 1$ e $y_{001} = 0$ e que $Q = 3000$, obtém-se as Equações (E.1), (E.2), (E.3) e (E.4).

$$S_{001} - S_{0-10} + 3000 \times (4 - x_{00} - x_{01} - y_{0-10} - y_{001}) \geq p_{0-10} \Leftrightarrow 3000 \geq 900 \tag{E.1}$$

$$S_{0-10} - S_{001} + 3000 \times (3 - x_{00} - x_{01} - y_{0-10} + y_{001}) \geq p_{001} \Leftrightarrow 706,43 \geq 0 \tag{E.2}$$

$$F_{001} - F_{0-10} + 3000 \times (4 - x_{00} - x_{01} - y_{0-10} - y_{001}) \geq p_{001} \Leftrightarrow 3000 \geq 900 \tag{E.3}$$

$$F_{0-10} - F_{001} + 3000 \times (3 - x_{00} - x_{01} - y_{0-10} + y_{001}) \geq p_{0-10} \Leftrightarrow 900 \geq 193,57 \tag{E.4}$$

No entanto, ao resolver as Equações (4.17), (4.18), (4.19) e (4.20), se $x_{00} = x_{01} = y_{0-10} = y_{001} = 1$, $S_{ilj} > 0$, $F_{ilj} > 0$ e, através do *array* auxiliar *full_p_3d*, $p_{001} = 515,87$, obtém-se as Equações (E.5), (E.6), (E.7) e (E.8).

$$S_{001} - S_{0-10} + 3000 \times (4 - x_{00} - x_{01} - y_{0-10} - y_{001}) \geq p_{0-10} \Leftrightarrow S_{001} \geq 900 \tag{E.5}$$

$$S_{0-10} - S_{001} + 3000 \times (3 - x_{00} - x_{01} - y_{0-10} + y_{001}) \geq p_{001} \Leftrightarrow S_{001} \leq 3190,56 \tag{E.6}$$

$$F_{001} - F_{0-10} + 3000 \times (4 - x_{00} - x_{01} - y_{0-10} - y_{001}) \geq p_{001} \Leftrightarrow F_{001} \geq 1415,87 \tag{E.7}$$

$$F_{0-10} - F_{001} + 3000 \times (3 - x_{00} - x_{01} - y_{0-10} + y_{001}) \geq p_{0-10} \Leftrightarrow F_{001} \leq 3706,43 \tag{E.8}$$

Isto confirma que $x_{00} = x_{01} = y_{0-10} = y_{001} = 1$ não infringe as Equações (4.17), (4.18), (4.19) e (4.20), portanto o facto de $y_{001} = 0$ deriva da influência de outra restrição, conjunto de restrições ou mesmo da função objetivo.

A função objetivo é, de facto, influenciada indiretamente pelo valor de y_{ilj}, ao controlar as variáveis F_{ilj} e S_{ilj}. Se $y_{001} = 0$ e $(l \neq j)$, os termos da função objetivo podem resumir-se aos termos da Equação (E.9).

$$\begin{aligned}\sum_{i=0}^{m-1}\sum_{l=-1}^{n-1}\sum_{j=0}^{n-1} F_{ilj} - S_{ilj} &= (F_{0-10} - S_{0-10}) + (F_{0-11} - S_{0-11}) + (F_{001} - S_{001}) + (F_{010} - S_{010}) \\ &= \mathbf{193,57}\end{aligned} \tag{E.9}$$

Se $y_{001} = 1$ e $(l \neq j)$, os termos da função objetivo podem resumir-se aos termos da Equação (E.10). Como a tarefa $j = 1$ não tem *delay* e não irão existir tempos de espera tendo em conta os *deadlines* em causa: $F_{001} = 1500$ e $S_{001} = 1500 - 515,87 = 984,13$.

$$\begin{aligned}\sum_{i=0}^{m-1}\sum_{l=-1}^{n-1}\sum_{j=0}^{n-1} F_{ilj} - S_{ilj} &= (F_{0-10} - S_{0-10}) + (F_{0-11} - S_{0-11}) + (F_{001} - S_{001}) + (F_{010} - S_{010}) \\ &= \mathbf{709,44}\end{aligned} \tag{E.10}$$

Como se pretende minimizar a função objetivo, é natural que seja escolhida $y_{001} = 0$, pois esta leva a um menor valor da função objetivo. Mais ainda, se não existissem as restrições das Equações (4.7), (4.8), (4.9) e (4.10), a variável y_{0-10} era nula e a solução teria um valor objetivo ótimo nulo. Como estas restrições são usadas apenas quando $l = -1$, é necessária outra restrição ou conjunto de restrições que garantam que, para qualquer $j = \{0, ..., n-1\}$, $y_{ilj} = 1$, independentemente dos índices i e l.

Ao adicionar a restrição de capacidade da Equação (5.4), a alocação do conjunto (2) a $m = 1$ robôs é conseguida nas 100 instâncias geradas, visto que se garante que $y_{0-10} = 1$ e $y_{001} = 1$ (Figura 5.8). No conjunto (2), se alterar a tarefa do tipo 2 (sem *delay*, nas Figuras 5.8a e 5.8b) para uma do tipo 4 (com *delay*), as duas tarefas são também alocadas em todas as 100 instâncias geradas, com e sem avaria (Figuras 5.8c e 5.8d).

A alocação do conjunto (2) a $m = 2$ e $m = 3$ robôs é igualmente conseguida (Figuras 5.9 e 5.10), com e sem *delay*. Para testar se são escolhidos 2 robôs diferentes quando as tarefas têm deadlines simultâneos, é antecipando o *deadline* da tarefa com *task_ID*= 2 para os 900 segundos. Verifica-se que, para $m = 2$ e $m = 3$, o conjunto (2) com *deadlines* simultâneos aloca corretamente todas as tarefas (Figura 5.11).

E.4 Iteração 4: Técnica de Reformulação-Linearização (RLT)

No conjunto (3), para além de ter mais tarefas, estas têm *deadlines* que requerem o sequenciamento de duas tarefas no mesmo robô, quando $m = 2$ (ou seja $n > m$). O caso $m = 1$, como existem tarefas com *deadlines* simultâneos, resulta num problema *infeasible* ou inválido.

Ao alocar as tarefas do conjunto (3) a $m = 2$ robôs, nem todas as alocações são feitas com sucesso, embora todas as instâncias sejam resolvidas e exista uma solução ótima a apresentar. Todas as alocações bem sucedidas ocorrem quando a tarefa $j = 1$ e $j = 2$ são alocadas ao mesmo robô e $j = 0$ é alocada a outro robô. Isto evita que exista sobreposição das tarefas $j = 0$ e $j = 2$ no mesmo robô. Seria de esperar que todas as outras instâncias repetissem este comportamento. No entanto, existe novamente incongruências entre as variáveis x_{ij} e y_{ilj} (Figura E.6).

O ficheiro *.lp*, tal como nos casos anteriores, indica que apenas algumas das variáveis $x_{ij}, y_{ilj}, p_{ilj}, S_{ilj}$ e C_{ilj} são necessárias para resolver o problema. As restantes foram reduzidas através dos métodos da rotina de *presolve*. Os ficheiros *.lp* de uma alocação correta e errada diferem apenas nas durações das tarefas e coeficientes derivados das mesmas. Mais ainda, todas as variáveis consideradas no sub-problema são iguais.

As Equações (4.17), (4.18), (4.19) e (4.20) são responsáveis pelo sequenciamento das tarefas, pois ($n > m$). É necessário recorrer novamente aos valores finais das variáveis e dos dois lados das restrições para identificar a origem da discrepância entre as variáveis. À semelhança das Equações (E.9) e (E.10), a alocação associada à Figura E.6 tem o valor ótimo da função objetivo de 561, 63 (Equação E.11).

$$\sum_{i=0}^{m-1}\sum_{l=-1}^{n-1}\sum_{j=0}^{n-1} F_{ilj} - S_{ilj} = (F_{0-10} - S_{0-10}) + (F_{012} - S_{012}) + (F_{1-11} - S_{1-11}) = \mathbf{561{,}63} \tag{E.11}$$

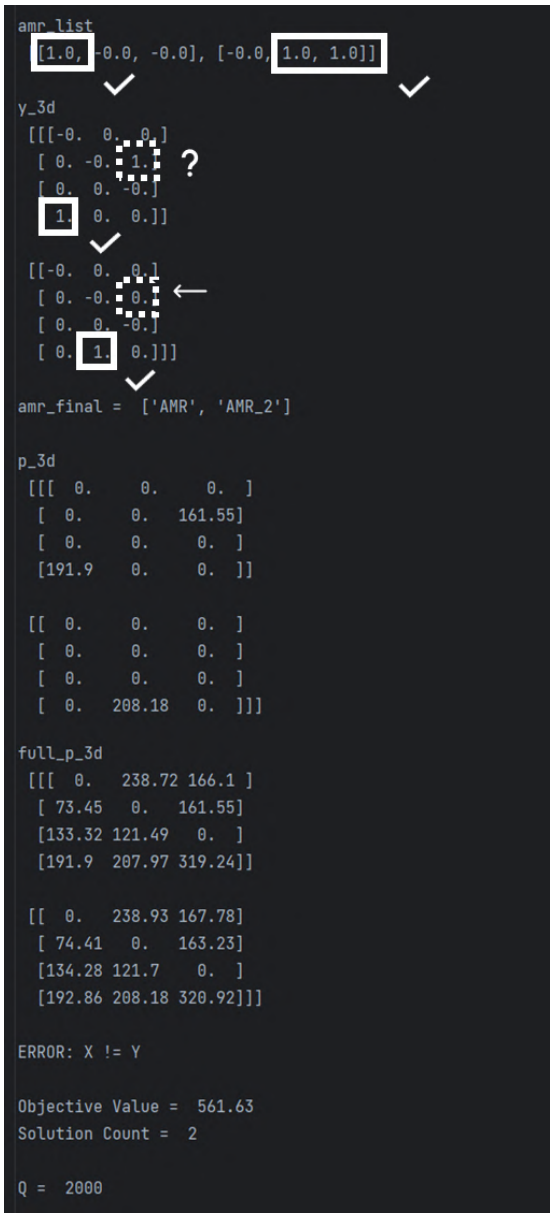


Figura E.6: Exemplo de instância onde o conjunto (3) não foi alocado corretamente a $m = 2$ robôs. A variáveis onde existem incongruências estão marcadas a tracejado. A seta indica que a variável $y_{112} = 0$ está errada. Mais ainda, se a tarefa $j = 2$ estivesse alocada ao robô $i = 0$, $y_{002} = 1$, e não $y_{012} = 1$.

O termo $(F_{012} - S_{012})$ não é o termo correto, visto que $y_{012} = 1$ está errado. No entanto, esta opção tem um valor ótimo da função objetivo menor do que a alocação correta (ou seja $y_{002} = 1$, na Equação E.12).

$$\sum_{i=0}^{m-1} \sum_{l=-1}^{n-1} \sum_{j=0}^{n-1} F_{ilj} - S_{ilj} = (F_{0-10} - S_{0-10}) + (F_{002} - S_{002}) + (F_{1-11} - S_{1-11}) = \mathbf{566{,}18} \tag{E.12}$$

Isto significa que as restrições de sequenciamento para $n > m$ não estão a impor os valores das variáveis x_{ij} e y_{ilj}. Qualquer combinação das variáveis x_{ij} e y_{ilj} não infringe as restrições, por isso a função objetivo determina quais os termos $(F_{ilj} - S_{ilj})$ diferentes de zero e, consequentemente, quais y_{ilj} são diferentes de zero. Como as restrições não são suficientemente rígidas para

determinar x_{ij}, esta variável toma o valor mais conveniente para o problema, quer esteja ou não de acordo com y_{ilj}, para quaisquer índices i, l e j.

A relação entre as variáveis x_{ij} e y_{ilj} é fulcral para a correta alocação das tarefas a cada robô. Para determinados índices i, l e j, apenas quando $y_{ilj} = 1$ é que $x_{ij} = 1$, o que nem sempre acontece. Se x_{ij} e y_{ilj} forem diferentes ou nulos, a tarefa j não é alocada ao robô i, após a tarefa l. Uma solução para este problema passaria por multiplicar as variáveis x_{ij} e y_{ilj}. No entanto, esta não é uma expressão linear, portanto não pode ser utilizada desta forma. É necessário transformar essa expressão na sua representação linear, se possível.

A *Reformulation-Linearization Technique* ou Técnica de Reformulação-Linearização (RLT) é uma técnica de linearização de problemas de Programação Inteira Mista (linear ou polinomial). É considerada uma técnica robusta para a representação linear deste tipo de problemas, embora acompanhada pelo aumento da complexidade do mesmo (consequência do aumento do número de restrições necessárias) (Pardalos et al., 2013, p. 2849).

O produto de variáveis $x_{ij} \times y_{ilj}$ é semelhante à proposição lógica *AND*, na qual é necessário que ambas as condições se verifiquem para que a conjunção das condições seja verdadeira. Esta multiplicação pode ser reformulada como uma variável binária z_{ilj}, cujo possíveis valores estão resumidos na Tabela E.1.

Tabela E.1: Possíveis resultados da multiplicação entre as variáveis x_{ij} e y_{ilj} (variável z_{ilj}).

x_{ij}	y_{ilj}	$z_{ilj} = x_{ij} \times y_{ilj}$
0	0	0
0	1	0
1	0	0
1	1	1

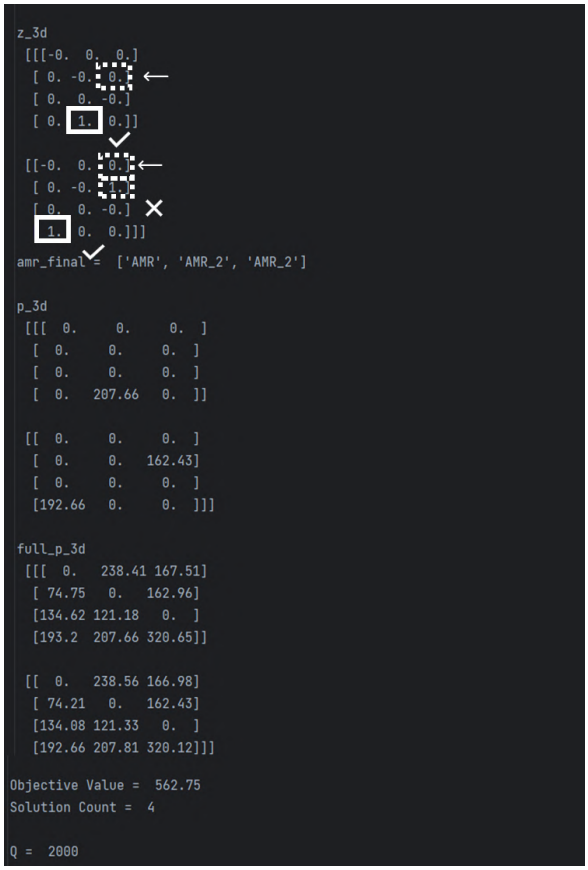
A variável z_{ilj} depende das variáveis x_{ij} e y_{ilj}, pelo que são necessárias restrições lineares para limitar o valor de z_{ilj}, tendo em conta as respetivas variáveis x_{ij} e y_{ilj} (Equações (5.5), (5.6), (5.7) e (5.8)).

Este conjunto de restrições lineares permite substituir os termos $(4 - x_{ij} - x_{ik} - y_{ilj} - y_{ijk})$ e $(3 - x_{ij} - x_{ik} - y_{ilj} + y_{ijk})$ nas Equações (4.17), (4.18), (4.19) e (4.20) pelos termos $(2 - z_{ilj} - z_{ijk})$ e $(1 - z_{ilj} + z_{ijk})$, respetivamente. De forma a garantir que, para cada tarefa j, existe uma e uma só variável z_{ilj} tal que $z_{ilj} = 1$, a Equação (5.4) é substituída pela Equação (5.9), que mantém os resultados positivos conseguidos pela restrição da Equação (5.4), na Iteração 3.

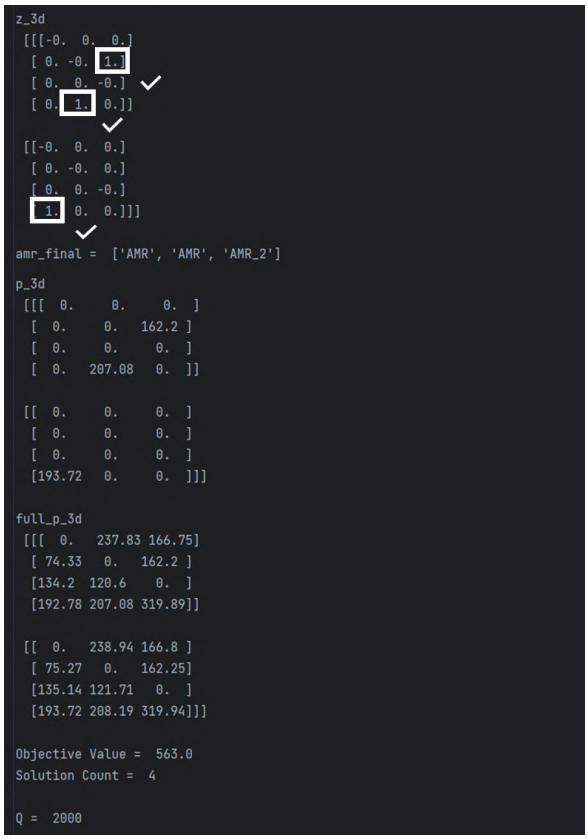
Após a implementação do conjunto de restrições, observa-se mais uma vez que, em 10 instâncias, existem algumas alocações corretas e outras onde existe sobreposição entre as tarefas $j = 1$ e $j = 2$ do conjunto (3) (exemplos na Figura E.7). Na Figura E.7a, a variável $z_{0-11} = 1$ e $z_{1-10} = 1$ estão corretas, visto que são as primeiras duas tarefas com *deadlines* mais próximos. Por outro lado, a variável $z_{112} = 1$ está incorreta por dois motivos:

1. Se $z_{1-10} = 1$, significa que a tarefa $j = 2$ ocorre após a tarefa $j = 0$, por isso a variável z_{112} deveria ser nula e $z_{102} = 1$;
2. Esta alocação está incorreta pois, se a tarefa $j = 2$ ocorrer depois da tarefa $j = 0$, existe sobreposição de tarefas no robô $i = 1$. A tarefa $j = 2$ deveria estar alocada ao robô $i = 0$

após a tarefa $j = 1$, por isso $z_{012} = 1$ e $z_{112} = z_{102} = 0$ (situação que ocorre na alocação correta, na Figura E.7b).



(a) Instância onde a alocação está errada



(b) Instância onde a alocação está correta

Figura E.7: Exemplo de uma instância gerada pela alocação das tarefas do conjunto (3) a $m = 2$ robôs.

Este facto é indicativo de que é a função objetivo é novamente a responsável pela escolha da sequência errada de tarefas (Cenário 5 na Tabela E.2). Consequentemente, as variáveis z_{ilj} são ativadas conforme necessário. Isto significa também que as restrições de sequenciamento não estão a garantir que não haja sobreposição, mas apenas que a soma das durações das tarefas em cada robô consegue ser inserida no intervalo de tempo previsto (Q). Este facto pode ser comprovado pelas restrições homólogas utilizadas por Ghaleb e Taghipour (2023), explicado no Apêndice F.

A alocação de $n = 3$ tarefas a $m = 2$ robôs implica que o problema *Mixed Integer Linear Programming* ou Programação Linear Inteira Mista (MILP) em questão seja composto por 379 restrições e 144 variáveis, tornando difícil a identificação de inconsistências com os resultados pretendidos. Embora a lista de variáveis e dos seus valores finais possam ser consultados através dos *arrays* construídos após a otimização, os lados esquerdo e direito de diferentes restrições não são facilmente comparáveis. De acordo com a forma standard indicada por Hillier e Lieberman (2015, p. 34), o *solver* permite consultar o lado esquerdo de cada restrição (*LHS*) através do parâmetro *getRow* (Gurobi Optimization, 2023a, p. 579), devolvendo uma expressão linear com as variáveis envolvidas e respetivos coeficientes (termos de ordem 1). Para este efeito, é necessário indicar a restrição que se pretende, ou seja, qual o elemento do *array* de restrições

Tabela E.2: Valores da função objetivo dependendo dos índices (i, l, j) para os quais $z_{ilj} = 1$. Para calcular o valor da função objetivo, é usado o *array full_p_3d* da alocação que originou o Cenário 5 (Figura E.7a). O Cenário 3 é aquele que deveria ocorrer nesta alocação em particular. Embora tenha um valor da função objetivo superior ao do Cenário 5, é aquele onde não existem sobreposições de tarefas e cujo valor da função objetivo é o menor dos cenário possíveis.

Cenário	$z_{ilj} = 1$	Valor da Função Objetivo (segundos)
1	$(0, -1, 0)$ $(1, -1, 1)$ $(0, 0, 2)$	**568,52**
2	$(0, -1, 0)$ $(1, -1, 1)$ $(1, 1, 2)$	**563,44**
3	$(0, -1, 1)$ $(1, -1, 0)$ $(0, 1, 2)$	**563,28**
4	$(0, -1, 1)$ $(1, -1, 0)$ $(1, 0, 2)$	**567,30**
5	$(0, -1, 1)$ $(1, -1, 0)$ $(1, 1, 2)$	**562,75**

lineares obtido pelo parâmetro *getConstrs*(Gurobi Optimization, 2023a, p. 568). Para consultar o lado direito de uma restrição (soma dos termos de ordem 0) é usado o atributo *RHS* (Gurobi Optimization, 2023a, p. 766). Para saber se a relação entre *LHS* e *RHS* é $\geq$, $\leq$ ou $=$ utiliza-se o atributo *Sense* (Gurobi Optimization, 2023a, p. 766). A conjugação destes parâmetros e atributos de cada restrição, obtém-se uma representação linear de cada restrição (Figura E.8). Juntamente com esta, pode consultar-se a diferença entre o lado direito e o lado esquerdo da restrição, após a otimização, usando o atributo *Slack* (Gurobi Optimization, 2023a, p. 768).

```
start1_1_1_0 = 0.0
start1_1_1_0: -100.0 y_1_1_0 + S_1_1_0 > 0.0

start1_1_2_0 = 0.0
start1_1_2_0: -100.0 y_1_2_0 + S_1_2_0 > 0.0

start1_1_-1_1 = -543.08
start1_1_-1_1: -100.0 y_1_-1_1 + S_1_-1_1 > 0.0
```

Figura E.8: Exemplo da representação de restrições lineares e respetiva folga como *output* do GRB. Cada restrição tem um nome, que incluí os índices i, l, j, k associados ao robô e tarefas em questão. A primeira linha de cada restrição é a folga ou *Slack* (por exemplo, a folga da restrição com o nome "*start1*"e índices $i = 1,\ l = -1,\ j = 1$ é de $-543,08$). A segunda linha da mesma restrição corresponde à representação linear da equação/inequação associada.

A representação das restrições lineares e da sua folga mediante estes parâmetros e atributos fornece alguma informação sobre as restrições de sequenciamento. Nomeadamente, pode confirmar que estas restrições são ativadas quando uma tarefa precede outra, no mesmo robô. A título de exemplo, são utilizadas as instâncias da Figura E.7, com especial atenção à instância

onde houve a alocação incorreta das tarefas (Figura E.7a) e em comparação com a instância onde a alocação foi feita corretamente (Figura E.7b). Será dada maior atenção ás Restrições de Sequenciamento das Equações Equações (4.17), (4.18), (4.19) e (4.20).

O *solver* calcula a folga de cada restrição segundo a expressão *Slack=RHS - LHS*. Dependendo se a relação entre *LHS* e *RHS* é $\geq$, $\leq$ ou $=$, a folga pode ser interpretada com a ajuda das Equaçãos (E.13), (E.14) e (E.15).

$$\text{Se LHS} \geq \text{RHS} \Leftrightarrow \text{RHS} - \text{LHS} = \textit{Slack} \leq 0 \tag{E.13}$$

$$\text{Se LHS} \leq \text{RHS} \Leftrightarrow \text{RHS} - \text{LHS} = \textit{Slack} \geq 0 \tag{E.14}$$

$$\text{Se LHS} = \text{RHS} \Leftrightarrow \text{RHS} - \text{LHS} = \textit{Slack} = 0 \tag{E.15}$$

Estas são as condições para que a restrição não seja infringida. Por exemplo, na Figura E.8, a restrição "start1_1_-1_1"tem uma folga negativa, pois, para que esta restrição seja respeitada, a *Slack* é sempre menor ou igual a zero. Assumindo que o *RHS* se mantém constante [1], o *LHS* pode diminuir 543, 08 unidades e a restrição iria ser igualmente respeitada.

As Restrições de Sequenciamento são inequações do tipo $\geq$, portanto a folga ou *Slack* será sempre ≤ 0 (Equação E.13). Quando $Slack = RHS$, significa que $LHS = 0$, como é o caso da restrição prec1_1_ − 1_1_0, na Figura E.9. Isto indica que todas as variáveis z_{ilj} e z_{ijk} são nulas e, consequentemente, p_{ilj}, S_{ilj} e S_{ijk} também, por influência das Restrições de Precedência. Na Figura E.9, as restrições prec1_1_ − 1_1_2 e prec1_0_ − 1_1_2 têm a mesma folga pois $z_{112} = 1$, o que leva a que S_{112} seja diferente de zero e igual ao seu valor final, após a otimização.

```
prec1_1_-1_1_0 = -4000.0
prec1_1_-1_1_0: -2000.0 z_1_-1_1 + -2000.0 z_1_1_0 + -1.0 p_1_-1_1 + -1.0 S_1_-1_1 + S_1_1_0 > -4000.0

prec1_1_-1_1_2 = -2887.57
prec1_1_-1_1_2: -2000.0 z_1_-1_1 + -2000.0 z_1_1_2 + -1.0 p_1_-1_1 + -1.0 S_1_-1_1 + S_1_1_2 > -4000.0

prec1_1_0_1_2 = -2887.57
prec1_1_0_1_2: -2000.0 z_1_0_1 + -2000.0 z_1_1_2 + -1.0 p_1_0_1 + -1.0 S_1_0_1 + S_1_1_2 > -4000.0
```

Figura E.9: Exemplo de *output* da restrição de sequenciamento "*prec1*"(Equação 4.17), associada à instância da Figura E.7a.

As restrições associadas ás Equações (4.18), (4.19) e (4.20) têm um comportamento muito semelhante. Na maioria das Restrições de Sequenciamento, $Slack = RHS$, o que indica que todas as variáveis envolvidas são nulas, para determinados índices i, l, j e k. Nos casos em que pelo menos uma das variáveis z_{ilj} ou z_{ijk} é igual a 1, $Slack \neq RHS$. No entanto este cenário apenas indica que uma tarefa está alocada e não estabelece nenhuma sequencia de tarefas. Quando $z_{ilj} = 1$ e $z_{ijk} = 1$, que indica a alocação de duas tarefas em sequência no mesmo robô, todas as variáveis são diferentes de zero e a restrição deverá manter a sua folga menor ou igual a zero para que o modelo seja viável.

[1] O *RHS* corresponde à soma dos termos de ordem 0 de uma restrição, ou seja, de parâmetros que se mantém constantes durante a otimização do problema

Ao analisar as 96 Restrições de Sequenciamento geradas, não existe nenhuma restrição que relacione $z_{1-10} = 1$ e $z_{112} = 1$ (*array* z_3d na Figura E.7), o que era de esperar pois, se $z_{1-10} = 1$, então z_{112} deveria ser nula e $z_{102} = 1$. Como $z_{112} = 1$ minimiza a função objetivo e não infringe qualquer uma das 96 Restrições de Sequenciamento (mantém todas as folgas iguais ou inferiores a zero), este caso é preferido. Isto faz com que não existam Restrições de Sequenciamento onde $z_{ilj} = 1$ e $z_{ijk} = 1$, o que indica, mais uma vez, a pouca influência que este tipo de restrições tem na prevenção de sobreposições entre tarefas, quando comparada com a influência da Função Objetivo.

A título de comparação, na alocação correta das tarefas (Figura E.7b), como as variáveis z já estão corretas, a sequência das tarefas $j = 1$ e $k = 2$ no robô $i = 0$ é reconhecida pelas Restrições de Sequenciamento. Tal como ilustrado na Figura E.10, existem quatro restrições onde $z_{0-11} = 1$ e $z_{012} = 1$. As restrições "prec1_0_ − 1_1_2"e "prec2_0_ − 1_1_2"complementam-se entre si, tal como as restrições "prec3_0_ − 1_1_2"e "prec4_0_ − 1_1_2".

```
prec1_0_-1_1_2 = -37.80000000000018
prec1_0_-1_1_2: -2000.0 z_0_-1_1 + -2000.0 z_0_1_2 + -1.0 p_0_-1_1 + -1.0 S_0_-1_1 + S_0_1_2 > -4000.0

prec2_0_-1_1_2 = -1592.9199999999998
prec2_0_-1_1_2: -2000.0 z_0_-1_1 + 2000.0 z_0_1_2 + -1.0 p_0_1_2 + S_0_-1_1 + -1.0 S_0_1_2 > -2000.0

prec3_0_-1_1_2 = -37.80000000000018
prec3_0_-1_1_2: -2000.0 z_0_-1_1 + -2000.0 z_0_1_2 + -1.0 p_0_1_2 + -1.0 F_0_-1_1 + F_0_1_2 > -4000.0

prec4_0_-1_1_2 = -1592.92
prec4_0_-1_1_2: -2000.0 z_0_-1_1 + 2000.0 z_0_1_2 + -1.0 p_0_-1_1 + F_0_-1_1 + -1.0 F_0_1_2 > -2000.0
```

Figura E.10: Exemplo de *output* da Restrições de Sequenciamento, associadas à instância da Figura E.7b. Para todas as restrições, $Q = 2000$ segundos. As restrições "prec1_0_ − 1_1_2"e "prec3_0_ − 1_1_2"estão influenciadas por um fator de 2, através do termo $(2 - z_{ilj} - z_{ijk})$, por isso é que o *RHS* é de −4000.

No diagrama da Figura E.11, podemos ver que, quando $z_{0-11} = 1$ e $z_{012} = 1$, as restrições garantem que existem tempo suficiente para alocar as tarefas $j = 1$ e $k = 2$ em sequência. As folgas representam o tempo no qual o robô está livre para executar outras tarefas ou dirigir-se ao parque de estacionamento. Este diagrama confirma também o resultado obtido no Apêndice F.

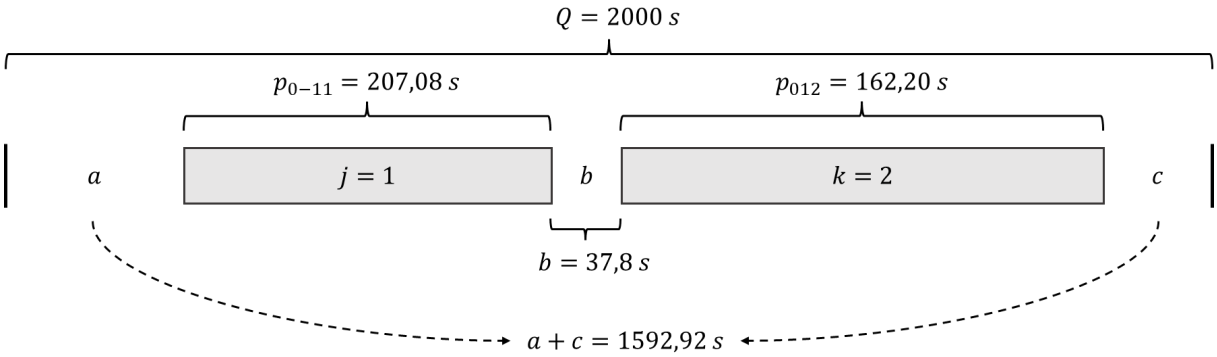


Figura E.11: Exemplo da relação entre as folgas das Restrições de Sequenciamento e a duração das tarefas sequenciadas $j = 1$ e $j = 2$ do conjunto (3), no robô $i = 0$. Os valores de a, b e c correspondem ás folgas do par de restrições "prec1_0_ − 1_1_2"e "prec2_0_ − 1_1_2", iguais ás folgas do par "prec3_0_ − 1_1_2"e "prec4_0_ − 1_1_2". A soma de todas as folgas e duração das tarefas corresponde ao *makespan* previsto Q, tal como esperado.

Novamente, isto não significa que as Restrições de Sequenciamento consigam evitar a sobreposição de tarefas no mesmo robô. Apenas confirmam se existe ou não tempo disponível no robô i para alocar determinada sequência de tarefas que minimize a Função Objetivo.

APÊNDICE F

RESTRIÇÕES DE SEQUENCIAMENTO

Ghaleb e Taghipour (2023) formulam um problema de Programação Linear Inteira Mista para realizar o planeamento robusto de ordens de produção de duração determinística que minimize os custos totais de produção e que seja resiliente a eventos imprevistos, como avarias de equipamentos durante a produção. Para o efeito, são utilizadas as restrições das Equações F.1 e F.2 para sequenciar as encomendas e garantir que não existem sobreposições de tarefas (tendo em conta que uma máquina satisfaz uma encomenda de cada vez).

Índices

- j Índice das encomendas $j = \{1, ..., n\}$
- k Índice das máquinas $k = \{1, ..., m\}$
- i Índice da posição de cada encomenda na sequência de uma máquina: $i = \{1, ..., n\}$

Parâmetros

- C_j Instante de conclusão da encomenda j
- p^B_{jk} Tempo de processamento da encomenda j na máquina k
- Q Número suficientemente grande, correspondente ao "makespan"previsto de todas as encomendas a calendarizar

Variáveis

- x_{jk} Variável binária que, quando $x_{jk} = 1$, indica que a encomenda j é produzida pela máquina k. Caso contrário, $x_{jk} = 0$
- y_{ijk} Variável binária que, quando $y_{ijk} = 1$, indica que a encomenda j é produzida pela máquina k antes da encomenda i. Caso contrário, $y_{ijk} = 0$

$$C_j - C_i + Q\,(2 - x_{jk} - x_{ik} + y_{ijk}) \geqslant p^B_{jk}, \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.1}$$

$$C_i - C_j + Q\,(3 - x_{jk} - x_{ik} - y_{ijk}) \geqslant p^B_{ik}, \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.2}$$

Quando $x_{jk} = x_{ik} = 1$ e $y_{ijk} = 1$, ao substituir na Equação F.1, obtém-se:

$$\begin{aligned} & C_j - C_i + Q \times (2 - x_{jk} - x_{ik} + y_{ijk}) \geqslant p^B_{jk} \\ \Leftrightarrow \quad & C_j - C_i + Q \times (2 - 1 - 1 + 1) \geqslant p^B_{jk} \\ \Leftrightarrow \quad & C_j - C_i + Q \geqslant p^B_{jk} \end{aligned} \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.3}$$

Da mesma forma, ao substituir na Equação F.2, obtém-se:

$$\begin{aligned} & C_i - C_j + Q \times (3 - x_{jk} - x_{ik} - y_{ijk})) \geqslant p^B_{ik} \\ \Leftrightarrow \quad & C_i - C_j + Q \times (3 - 1 - 1 - 1) \geqslant p^B_{ik} \\ \Leftrightarrow \quad & C_i - C_j \geqslant p^B_{ik} \end{aligned} \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.4}$$

É possível relacionar as Equações F.3 e F.4, utilizando a Equação F.5.

$$\begin{aligned} & C_j - C_i = -(C_i - C_j) \leqslant -p^B_{ik} \\ \Leftrightarrow \quad & C_i - C_j \geqslant p^B_{ik} \end{aligned} \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.5}$$

Substituindo a Equação F.5 na Equação F.3:

$$\begin{aligned} & C_j - C_i + Q \geqslant p^B_{jk} \\ \Leftrightarrow \quad & Q \geqslant p^B_{jk} + (C_i - C_j) \\ \Leftrightarrow \quad & Q \geqslant p^B_{jk} + p^B_{ik}, \quad \text{pois } C_i - C_j \geqslant p^B_{ik} \end{aligned} \qquad \begin{matrix}\forall j, i = \{1, \ldots, n\} \\ (j < i)\end{matrix} \tag{F.6}$$

Através da Equação F.6 garante-se que o intervalo de tempo Q é suficiente para calendarizar duas encomendas consecutivas j e i. Como se trata de um problema de planeamento de produção, esta relação é feita através dos instantes de conclusão de cada encomenda C_j, não existindo intervalos entre duas encomendas (*setup times* de cada máquina já estão incluídos nos tempos de processamento p^B_{ik}). Sendo assim, considera-se o instante de conclusão de uma encomenda C_j é o instante de início da encomenda seguinte (C_i). Isto facilita a utilização deste tipo de restrições para sequenciar as encomendas no planeamento, juntamente com outras restrições que limitam o instante de conclusão ao somatório dos tempos de processamento das encomendas anteriores á encomenda j.