

Centro Federal de Educação Tecnológica de Minas Gerais
Programa de Pós-graduação em Modelagem Matemática e Computacional

Modelos e algoritmos para problemas de planejamento de ordens de manutenção preventiva de longo prazo

João Luiz Marques de Andrade

Belo Horizonte
Outubro de 2025

João Luiz Marques de Andrade

Modelos e algoritmos para problemas de ordens de manutenção preventiva de longo prazo

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, como parte dos requisitos necessários para obtenção do título de Doutor em Modelagem Matemática e Computacional.

Área de Concentração: Modelagem Matemática e Computacional

Linha de Pesquisa: Sistemas Inteligentes

Orientador: Prof. Dr. Marcone Jamilson Freitas Souza

Coorientadores: Profa. Dra. Elisangela Martins de Sá
Prof. Dr. Gustavo Campos Menezes
Prof. Dr. Sérgio Ricardo de Souza

Belo Horizonte - MG
Outubro de 2025

Andrade, João Luiz Marques de
A554m Modelos e algoritmos para problemas de ordens de manutenção preventiva de longo prazo / João Luiz Marques de Andrade. – 2025.
119 f.

Tese (doutorado) apresentada ao Programa de Pós-Graduação em Modelagem Matemática e Computacional.
Orientador: Marcone Jamilson Freitas Souza.
Coorientadores: Elisangela Martins de Sá, Gustavo Campos Menezes, Sérgio Ricardo de Souza.
Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas Gerais.

1. Otimização matemática – Teses. 2. Equipamentos industriais – Manutenção e reparo – Gestão – Teses. 3. Espaço e tempo – Modelos matemáticos – Teses. 4. Pesquisa operacional – Teses. 5. Metaheurística – Teses. 6. Matheurística – Teses. 7. Habilidades – Teses. I. Souza, Marcone Jamilson Freitas. II. Sá, Elisangela Martins de. III. Menezes, Gustavo Campos. IV. Souza, Sérgio Ricardo de. V. Centro Federal de Educação Tecnológica de Minas Gerais. VI. Título.

CDD 519.6

Elaboração da ficha catalográfica pela bibliotecária Jane Marangon Duarte, CRB 6ª 1592 / Cefet/MG

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
COORDENAÇÃO DO PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM MATEMÁTICA E COMPUTACIONAL

"MODELOS E ALGORITMOS PARA PROBLEMAS DE PLANEJAMENTO DE ORDENS DE MANUTENÇÃO PREVENTIVA DE LONGO PRAZO".

Tese de Doutorado apresentada por **João Luiz Marques de Andrade,** em 27 de outubro de 2025, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Documento assinado digitalmente
gov.br
MARCONE JAMILSON FREITAS SOUZA
Data: 29/10/2025 12:16:09-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Marcone Jamilson Freitas Souza (Orientador)
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
GUSTAVO CAMPOS MENEZES
Data: 30/10/2025 08:18:11-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gustavo Campos Menezes (Coorientador)
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
ELISANGELA MARTINS DE SA
Data: 30/10/2025 09:01:02-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Elisangela Martins de Sá (Coorientadora)
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
SERGIO RICARDO DE SOUZA
Data: 30/10/2025 09:24:02-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Sérgio Ricardo de Souza (Coorientad
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
LUCIANO PERDIGAO COTA
Data: 29/10/2025 13:31:48-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Luciano Perdigão Cota
Instituto Tecnológico Vale

Documento assinado digitalmente
gov.br
BRUNO FERREIRA ROSA
Data: 29/10/2025 15:29:18-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Bruno Ferreira Rosa
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
DOUGLAS ALEXANDRE GOMES VIEIRA
Data: 29/10/2025 18:25:30-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Douglas Alexandre Gomes Vieira
Centro Federal de Educação Tecnológica de Minas Gerais

Documento assinado digitalmente
gov.br
ADRIANO CHAVES LISBOA
Data: 29/10/2025 19:12:46-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Adriano Chaves Lisboa
Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida à impressão,

Prof. Dr. Alisson Marques da Silva
Coordenador do Programa de Pós-Graduação Stricto Sensu em Modelagem Matemática e Computacional

Agradecimentos

É difícil expressar em palavras tudo o que senti nesses últimos 4 anos. Não são muitos que conhecem ou já vivenciaram o quão difícil é obter o título de doutor. Eu acredito que não existe doutorado fácil. Então, durante esses 4 anos, eu sempre tentei dar o meu máximo possível, porque assim pelo menos eu havia tentado. Pois bem, eu consegui! No entanto, para chegar até aqui, eu tive ajuda e apoio de muitas pessoas, então eu gostaria que cada uma delas se sentisse imensamente agradecida.

Primeiramente, eu gostaria de agradecer aos meus pais, Neusa e Amaury, e à Raphaela. Eles sempre estiveram ao meu lado, apoiando as minhas decisões e me ajudando absolutamente em tudo que precisei, seja rezando, ascendendo muitas velas, seja por me ajudar financeiramente, e por acreditar que eu era capaz.

Esse doutorado só ocorreu graças à oportunidade dada pelo Marcone e pelo Sérgio. Vocês dois são profissionais sensacionais, em que é muito nítido o vasto conhecimento e experiência que vocês têm sobre o assunto. Isso é admirável. Eu agradeço por confiar em mim, no meu trabalho e, principalmente, por me orientar.

Para Elisangela e Gustavo, o mais profundo agradecimento. Para ser mais sincero, eu não sei como agradecer. O mestrado e o doutorado só foram possíveis graças ao e-mail que a Elisangela respondeu lá em 2017. Então, sem ela, nada disso seria possível. Infelizmente, não deu certo em um primeiro momento e, com isso, ela me apresentou o Gustavo. Ele me deu a oportunidade de fazer o mestrado. Essa oportunidade foi algo muito importante para mim, em que eu me dediquei ao máximo para aproveitá-la. Gustavo, você é uma pessoa extraordinária. Eu agradeço por cada ideia, orientação, sugestão e palavra sua, só eu sei o quanto elas me ajudaram nessa caminhada. Conversar com você sempre é muito bom. Esse doutorado só foi possível de ser concluído graças às primeiras orientações que você me passou. Elisangela, você é brilhante e admirável. Muita coisa que sei sobre Pesquisa Operacional é graças às suas aulas e explicações. Você e o Gustavo me ensinaram a gostar de Pesquisa Operacional. Então, professora, eu agradeço a ajuda, dedicação, paciência, orientações, sugestões e tudo mais, porque elas foram essenciais.

Espero continuar a parceria com vocês e que um dia eu possa ser um orientador tão bom quanto vocês foram para mim. Eu sinto muito orgulho de ser aluno de vocês.

Resumo

Esta tese pesquisa dois problemas de planejamento de manutenções preventivas. O primeiro é o problema de planejamento de ordens de manutenção preventiva de longo prazo (LTPMOSP). Ele consiste em decidir simultaneamente quais ordens de manutenção devem ser executadas, bem como fazer a atribuição das ordens de manutenção às equipes e às máquinas e determinar quando cada ordem de manutenção deve ser executada ao longo do horizonte de planejamento. Cada manutenção requer apenas uma habilidade das equipes para ser executada. O segundo problema é uma extensão do LTPMOSP, em que as manutenções podem requerer uma ou mais habilidades para serem executadas, ou seja, pode-se ter manutenções mono ou multi-habilidade. A incorporação disso origina o problema de planejamento de ordens de manutenção preventiva de longo prazo com multi-habilidades (LTPMOSP-MS). Ambos os problemas têm o objetivo de minimizar o número de equipes utilizadas no planejamento e a soma das penalidades das ordens de manutenção não executadas. Para solucionar o LTPMOSP, são propostas duas novas formulações matemáticas indexadas no tempo. Além disso, para solucionar as instâncias de grande porte, propõe-se um algoritmo baseado na meta-heurística GRASP. Os resultados dos experimentos computacionais mostraram que as novas formulações, resolvidas via o resolvedor de programação matemática CPLEX, superaram significativamente a formulação da literatura, sendo capazes de encontrar novas soluções ótimas para instâncias com até 600 ordens de manutenção. Os resultados dos experimentos também mostraram uma superioridade do algoritmo GRASP em relação aos métodos heurísticos da literatura, gerando melhores limites superiores em um tempo computacional expressivamente menor. Para solucionar o LTPMOSP-MS, foram propostas uma formulação matemática indexada no tempo e três versões diferentes de uma nova heurística de duas fases (H2F). A primeira fase da heurística consiste em atribuir manutenções às equipes, enquanto a segunda fase programa as manutenções nas equipes respeitando todas as restrições do LTPMOSP-MS. Os experimentos computacionais mostraram que uma versão da H2F conseguiu encontrar limites inferiores e superiores melhores do que os que o resolvedor CPLEX encontra ao resolver a formulação proposta para o LTPMOSP-MS para instâncias com até 80 manutenções. Para instâncias com 150 e 300 manutenções, o CPLEX não encontrou nenhuma solução viável ao resolver a formulação proposta, enquanto a versão da H2F com pseudo-corte conseguiu encontrar a solução ótima para 50% das instâncias.

Palavras-chaves: Planejamento de manutenção de longo prazo, Formulação matemática indexada no tempo, GRASP, Meta-heurística, Matheurísticas, Multi-habilidade.

Abstract

This thesis investigates two preventive maintenance planning problems. The first is the long-term preventive maintenance order planning problem (LTPMOSP). It consists of simultaneously deciding which maintenance orders should be executed, as well as assigning maintenance orders to teams and machines and determining when each maintenance order should be executed throughout the planning horizon. Each maintenance requires only one skill to be executed, i.e., maintenance is mono-skilled. The second problem is an extension of LTPMOSP, in which maintenance may require one or more skills to be executed, i.e., there may be mono- or multi-skilled maintenance. The incorporation of this gives rise to the long-term preventive maintenance order planning problem with multi-skills (LTPMOSP-MS). Both problems aim to minimize the number of teams used in planning and the sum of the penalties for maintenance not scheduled. To solve LTPMOSP, two new mathematical formulations are proposed. In addition, to solve large instances, an algorithm based on the GRASP metaheuristic is proposed. The results of the computational experiments show that new formulations, solved via solver CPLEX, significantly outperform the formulation in the literature, being able to find new optimal solutions for instances with up to 600 maintenance orders. The results of the experiments also show that the GRASP algorithm is superior to the heuristic methods in the literature, generating better upper bounds in a significantly shorter computational time. To solve LTPMOSP-MS, a mathematical formulation and three different versions of a new two-phase heuristic (2PH) are proposed. The first phase of the heuristic consists of assigning maintenance to teams, while the second phase schedules maintenance in teams, respecting all constraints of the LTPMOSP-MS. Computational experiments show that a version of 2PH was able to find better lower and upper bounds than those found by solver CPLEX when solving the proposed formulation for LTPMOSP-MS for instances with up to 80 maintenances. For instances with 150 and 300 maintenances, the solver did not find any feasible solution when solving the proposed formulation, while the pseudo-cut version of 2PH was able to find an optimal solution for 50% of the instances.

Keywords: Long-term maintenance scheduling, Time-indexed formulations, GRASP, Metaheuristic, Matheuristic, Multi-skill

Sumário

Lista de Tabelas x
Lista de Figuras xi
Lista de Símbolos xiii
Lista de Algoritmos xiii

1 Introdução **1**
1.1 Apresentação Geral 1
1.2 Objetivos 5
1.2.1 Objetivos específicos 5
1.3 Contribuições 6
1.4 Organização do trabalho 8

2 Revisão de Literatura **9**
2.1 Problemas de Sequenciamento de Produção 9
2.1.1 Literatura relacionada 13
2.1.2 Formulações matemáticas 15
2.2 Otimização de planejamento de manutenções preventivas 22
2.3 Problemas de sequenciamento com multi-habilidades 26
2.4 Problemas de planejamento de manutenções preventivas com multi-habilidades 27
2.5 Métodos heurísticos 28
2.5.1 *Greedy randomized adaptative search procedure* (GRASP) . . . 28
2.6 Considerações parciais 30

3 Problema de planejamento de ordens de manutenção de longo prazo **31**
3.1 Descrição do problema 31
3.2 Exemplo para descrição do LTPMOSP 32
3.3 Formulação matemática da literatura 34
3.4 Formulação baseada em variáveis indexadas no tempo 36
3.5 Formulação baseada em variáveis indexadas no tempo e variáveis de precedência 38
3.6 Análise e comparação das dimensões dos modelos 44
3.7 Métodos de solução propostos baseados no algoritmo GRASP 45
3.7.1 Método heurístico GRASP 46
3.7.2 Método exato híbrido 54
3.8 Experimentos computacionais em instâncias do LTPMOSP 54
3.8.1 Instâncias 55
3.8.2 Comparação dos métodos exatos 55

3.8.3 Comparação entre as soluções heurísticas 63
3.9 Conclusões parciais 67

4 Problema de planejamento de ordens de manutenção de longo prazo com multi-habilidade 69
4.1 Descrição do problema LTPMOSP-MS 69
4.1.1 Um exemplo numérico 71
4.2 Formulação Matemática do LTPMOSP-MS 72
4.3 Heurística de duas fases 74
4.3.1 Formulação matemática para a Fase 1 da H2F 76
4.3.2 Formulação matemática para a fase 2 da H2F 81
4.3.3 Heurística de duas fases com decomposição do problema na segunda fase 82
4.3.4 Heurística de duas fases com um único problema na segunda fase 83
4.3.5 Heurística de duas fases iterativa 84
4.4 Experimentos Computacionais para a solução do LTPMOSP-MS . . . 86
4.4.1 Experimentos do conjunto de instâncias $\mathcal{P}$ 86
4.4.2 Experimentos do conjunto de instâncias $\mathcal{M}$ 94
4.5 Conclusões parciais para o estudo do LTPMOSP-MS 98

5 Considerações finais e trabalhos futuros 100
5.1 Considerações finais 100
5.2 Trabalhos derivados desta pesquisa 102
5.3 Trabalhos futuros 102

Referências 104

Apêndices 117

A Complexidade do subproblema de posicionamento de manutenção 118

Lista de Tabelas

2.1 Entradas para o campo α. 11
2.2 Entradas para o campo β. 11
2.3 Entradas para o campo γ. 12
2.4 Dimensões das formulações do problema $P_m||\sum C_j$. 22
2.5 Resumo das características comuns dos problemas abordados nos artigos citados 25

3.1 Dados de entrada das tarefas. 33
3.2 Dados de entrada das equipes de trabalho. 33
3.3 Dimensões das formulações. 45
3.4 Características das instâncias. 55
3.5 Resultados dos experimentos computacionais utilizando a formulação NET para as instâncias do conjunto $\mathcal{P}$. 56
3.6 Resultados dos experimentos computacionais utilizando a formulação TI para as instâncias do conjunto $\mathcal{P}$. 56
3.7 Resultados dos experimentos computacionais utilizando a formulação TIP para as instâncias do conjunto $\mathcal{P}$. 57
3.8 Resumo dos resultados computacionais das formulações para as instâncias do conjunto $\mathcal{P}$. 57
3.9 Parâmetros, conjunto de valores fornecidos ao IRACE e valores retornados por ele para instâncias do conjunto $\mathcal{P}$. 60
3.10 Resultados computacionais de cada método GRASP+Formulação para as instâncias do conjunto $\mathcal{P}$. 60
3.11 Resultados computacionais de cada método GRASP+Formulação para as instâncias do conjunto $\mathcal{M}$. 61
3.12 Resultados computacionais para as instâncias do conjunto $\mathcal{M}$. 62
3.13 Conjunto de valores fornecidos ao IRACE e valores retornados por ele para cada parâmetro do GRASP aplicado ao conjunto $\mathcal{G}$. 64
3.14 Conjunto de valores fornecidos ao IRACE e valores retornados por ele para cada parâmetro do GRASP aplicado ao conjunto $\mathcal{GG}$. 64
3.15 Resultados providos pelo algoritmo GRASP nas instâncias dos conjuntos M, G e GG. As 66

4.1 Dados de entrada das manutenções. 71
4.2 Dados de entrada das equipes. 72
4.3 Resultados computacionais para a formulação $\mathcal{TI}$ e para as diferentes versões da H2F para solucionar instâncias do conjunto $\mathcal{P}$. 89

4.4 Valores médios de tempo de execução e porcentagem de manutenções programadas de todos os métodos de solução para cada valor de $|I|$ e Ω. 92
4.5 Valores médios de tempo de execução e porcentagem de manutenções programadas de todos os métodos de solução para cada valor de $|M|$ e Φ. 93
4.6 Características das instâncias do conjunto $\mathcal{M}$ 95
4.7 Resultados computacionais para a formulação $\mathcal{TI}$ e as diferentes versões da H2F para solucionar instâncias do conjunto $\mathcal{M}$. 96

Lista de Figuras

3.1 Ilustração de uma solução factível para o LTPMOSP. 34
3.2 Exemplos de soluções para as variáveis Δ_{ik}^{t}. 39
3.3 Exemplo de solução com duas manutenções alocada em uma equipe. . 40
3.4 Exemplo de solução com três manutenções, associadas à mesma máquina, alocadas em duas equipes. 40
3.5 Exemplo ilustrando os parâmetros δ_i^{α} e δ_i^{β} para a janela de tempo da manutenção i. 41
3.6 Gráfico de barra com *gaps* gerados pelas formulações para as instâncias do conjunto $\mathcal{P}$. 59
3.7 Figura com os melhores gaps obtidos pelos métodos das formulações NET e TIP para as instâncias do conjunto $\mathcal{M}$. 63

4.1 Possibilidades de JRs para as manutenções. 71
4.2 Ilustração de uma solução factível para o LTPMOSP-MS, representando a atribuição simultânea das manutenções a equipes e máquinas, assim como o tempo de início e fim das manutenções programadas. . 72
4.3 Sobreposição das JRs das manutenções 1 e 2. 77
4.4 JS considerando as janelas das manutenções 1 e 2. 78
4.5 Ilustração de uma JS associada às manutenções 1 e 2 e das JRs de cinco manutenções. 78
4.6 Valores médios de porcentagem de manutenções programadas e equipes utilizadas no planejamento para as diferentes versões da H2F. 98

Lista de Siglas

LTPMOSP	*Long-Term Preventive Maintenance Order Scheduling Problem*
OASP	*Order Acceptance and Scheduling Problem*
PLI	Programação Linear Inteira
GRASP	*Greedy Randomized Adaptive Search Procedures*
LC	Lista de Candidatos
LRC	Lista Restrita de Candidatos
PM	Posicionamento de Manutenções
ILS	*Iterated Local Search*
NET	Formulação baseada em Variáveis de Rede
TI	Formulação baseada em Variáveis Indexadas no Tempo (LTPMOSP)
TIP	Formulação baseada em Variáveis Indexadas no Tempo e de Precedência
RPD	*Relative Percentage Deviation*
LTPMOSP-MS	*Long-Term Preventive Maintenance Order Scheduling Problem with Multi-Skill*
JR	Janela Resultante (LTPMOSP-MS)
$\mathcal{TI}$	Formulação baseada em Variáveis Indexadas no Tempo (LTPMOSP-MS)
H2F	Heurística de Duas Fases
$\mathcal{F}_I$	Formulação da Fase 1 da H2F
$\mathcal{F}_I I$	Formulação da Fase 2 da H2F
H2F-hor	Heurística de duas fases com um único problema na segunda fases
H2F-per	Heurística de duas fases com decomposição do problema na segunda fase
H2F-ite	Heurística de duas fases iterativa

Lista de Algoritmos

1 Meta-heurística GRASP 29

2 Algoritmo PM 47
3 Função *encontraIntervalo(.)* 49
4 Algoritmo *GRASP* proposto 51
5 Função $faseConstrucao(\alpha)$ 52
6 Método exato híbrido 54

7 Algoritmo da H2F-per 83
8 Algoritmo da H2F-hor 84
9 Algoritmo da H2F-iter 85

Capítulo 1

Introdução

Este capítulo apresenta uma breve introdução à tese. Inicialmente, a Seção 1.1 descreve o contexto no qual a tese está inserida. Em seguida, a Seção 1.2 detalha os objetivos desta pesquisa, enquanto que a Seção 1.3 expõe suas principais contribuições. Por fim, a Seção 1.4 descreve a estrutura do trabalho.

1.1 Apresentação Geral

A manutenção de equipamentos é de grande importância para empresas dos mais diversos setores. Na indústria de mineração não é diferente. Os ativos de uma mina (caminhões, tratores, escavadeiras, britadores, dentre outros) são caros e essenciais para o sucesso da exploração mineral. No entanto, eles estão sujeitos à degradação devido a, por exemplo, sua utilização, sua exposição a fatores climáticos ou outros fatores. Esta degradação pode resultar em uma falha do sistema. Uma parada não programada por falta de manutenção dos equipamentos causa atraso na produção e aumento de custos. A manutenção preventiva tem papel fundamental para garantir a vida útil dos ativos de mineração, reduzir custos e aumentar a competitividade.

Embora seja rotina para muitas organizações programar a manutenção preventiva dos equipamentos, levando em consideração equipes de trabalho, ordens de manutenção, janelas de tempo de inatividade dos equipamentos e o alto e diverso número de ativos de uma mina, se trata, contudo, em uma tarefa complexa, cara e desafiadora.

Esta tese aborda dois problemas de planejamento de manutenção preventiva. O primeiro é o Problema de Planejamento de Ordens de Manutenções Preventivas de Longo Prazo (*Long-Term Preventive Maintenance Order Scheduling Problem* - LTPMOSP). Ele foi introduzido por Aquino et al. (2018a) para resolver um problema da indústria de mineração de ferro. O LTPMOSP busca programar um conjunto de manutenções preventivas para um horizonte de 52 semanas, que é a razão pela qual

é considerado um problema de planejamento de longo prazo. Segundo os autores, o planejamento anual das manutenções, conhecido como mapa ou plano de 52 semanas, era gerado por um *software* que não considerava as habilidades e disponibilidades das equipes para executá-las. Quando cada gerente de área de manutenção recebia as programações, ele/ela as atribuía às suas equipes sem considerar a disponibilidade das equipes das outras áreas. Como resultado, as manutenções frequentemente se sobrepunham, e era possível concluir aproximadamente 50% dos trabalhos de manutenção (Aquino et al., 2019). Assim, para cumprir o cronograma proposto, equipes terceirizadas tiveram que ser contratadas, tornando os custos de manutenção mais caros.

O LTPMOSP envolve um conjunto de máquinas que devem ser submetidas a um conjunto de ordens de manutenção preventiva e um conjunto de equipes de trabalho com diferentes habilidades para executá-las. Além disso, cada manutenção está associada a uma janela de tempo para executá-la. O problema consiste em decidir, simultaneamente, quais ordens de manutenção devem ser executadas, quantas equipes serão utilizadas, atribuir as ordens às equipes e determinar quando cada ordem de manutenção deve ser executada. A decisão deve levar em conta que:

(a) a manutenção só é executada por uma equipe que possua habilidade que ela requer;

(b) se a manutenção for programada, ela deve ser programada dentro da sua janela de tempo e da disponibilidade da equipe que executá-la;

(c) uma equipe pode executar apenas uma manutenção por vez;

(d) as manutenções programadas em uma mesma máquina não podem se sobrepor;

(e) se a manutenção não for programada, uma penalidade é incorrida.

O objetivo do LTPMOSP é minimizar o número de equipes utilizadas no planejamento e a soma das penalidades das manutenções não programadas. De acordo com Aquino et al. (2019), o objetivo de ter o maior número de ordens planejadas para serem executadas está associado à importância das manutenções preventivas. Por outro lado, o menor número de equipes utilizadas é justificado pela busca de redução de custos.

No LTPMOSP, cada ordem de manutenção tem uma duração pré-determinada, semelhante aos tempos de processamento das tarefas nos problemas de sequenciamento de máquinas (Pinedo, 2022). É possível fazer uma analogia entre o LTPMOSP e problemas de sequenciamento em máquinas, em que as ordens, que devem ser executadas por equipes, podem ser visualizadas como tarefas que devem ser processadas

por máquinas, demonstrando, então, uma proximidade entre ambos os tipos de problemas. Além disso, o LTPMOSP também possui conexão com os problemas de *Crew Scheduling* (Kasirzadeh et al., 2017) e *Surgical Scheduling* (Silva et al., 2015), pois cada equipe possui uma habilidade e a resolução do problema gera uma programação para as equipes.

Contudo, a conexão com o problema investigado em Silva et al. (2015) é ainda maior. Para esse problema, pode-se fazer uma analogia entre as cirurgias e as manutenções; os cirurgiões e as equipes; e as salas de cirurgias e as máquinas. De forma similar ao LTPMOSP, em que cada equipe pode executar apenas uma manutenção e que pode haver somente uma manutenção executada em uma máquina, no problema de Silva et al. (2015) um cirurgião pode realizar apenas uma cirurgia por vez e pode-se ter apenas uma cirurgia ocorrendo por vez em cada sala. Outra semelhança entre esses problemas é em relação à janela de tempo. As cirurgias têm uma janela de tempo para serem realizadas, assim como as manutenções. Além disso, no LTPMOSP essa janela é afetada pela disponibilidade da equipe; já no problema de Silva et al. (2015), ela é afetada pela disponibilidade dos cirurgiões.

Aquino et al. (2018a) mostram que o LTPMOSP é $\mathcal{NP}$-difícil, uma vez que este problema é um caso particular do problema de sequenciamento de uma única máquina, que é $\mathcal{NP}$-difícil (Brucker, 2004; Brucker & Knust, 2009). Devido a isso, esses autores justificam o desenvolvimento de uma formulação matemática e de algoritmos meta-heurísticos para solucionar o LTPMOSP. Embora eles tenham propostos uma formulação matemática, os trabalhos citados apresentam um foco maior na abordagem heurística de solução, bem como em abordagens que geram um único planejamento para o longo horizonte.

Esta tese, no entanto, avança em uma linha de pesquisa diferente, apresentando um foco maior em métodos de solução exatos, com a proposição de novas formulações para o problema em estudo. Dada a forte conexão do LTPMOSP e os problemas de sequenciamento de máquinas, Aquino et al. (2018a) propuseram uma formulação matemática baseada em variáveis em rede (Queyranne & Schulz, 1994). Esta classe de variável tem, como vantagem, o baixo tempo de resolução da relaxação linear, porém, tem como desvantagem o fornecimento de limites inferiores fracos (no caso de problemas de minimização) (Unlu & Mason, 2010). Diante desta desvantagem, esta tese propõe novas formulações que geram limites inferiores mais fortes e que, consequentemente, são mais eficientes em encontrar soluções ótimas para o problema.

Uma vez que os métodos exatos são impraticáveis para instâncias de grande porte, esta tese também propõe um algoritmo meta-heurístico para o LTPMOSP. O algoritmo proposto é baseado na meta-heurística *Greedy Randomized Adaptive Search Procedures* (GRASP) (Feo & Resende, 1989, 1995). O GRASP é uma meta-heurística

consolidada, que resolveu com sucesso muitos problemas de otimização combinatória, como em Carrabs et al. (2024), Nadali et al. (2024), Cheimanoff et al. (2023), Pérez et al. (2023), Yepes-Borrero et al. (2023), Souza Almeida et al. (2022) e Vu et al. (2022). A formulação do GRASP apresentada nessa tese utiliza um novo algoritmo de posicionamento de manutenções. Ele determina o início de cada manutenção e a equipe que vai executá-la. Por fim, é apresentado um método exato híbrido que combina as formulações matemáticas com o algoritmo GRASP. Este método utiliza o algoritmo GRASP para gerar uma solução inicial, que é utilizada pelo *solver* para resolver as formulações matemáticas.

O segundo problema estudado nesta tese é o Problema de Planejamento de Manutenções Preventivas com Multi-habilidades (*Long-Term Preventive Maintenance Order Scheduling Problem with Multi-Skill* – LTPMOSP-MS). Esse problema é introduzido originalmente nessa tese. Ele é uma extensão do LTPMOSP, que incorpora manutenções com multi-habilidades. Essas manutenções são as que requerem duas ou mais habilidades para serem executadas. Elas podem ser vistas como um conjunto de manutenções que podem ser executadas ao mesmo tempo. A execução de várias manutenções ao mesmo tempo faz com que a produção tenha menos paradas. No LTPMOSP são consideradas apenas manutenções com mono-habilidade, ou seja, todas as manutenções requerem apenas uma única habilidade para serem executadas. No problema estudado nesta tese, são consideradas manutenções que requerem mono e multi-habilidades. Com essa modificação, as manutenções com multi-habilidades requerem mais de uma equipe para serem executadas, uma vez que cada equipe possui apenas uma habilidade.

Outra modificação incorporada ao problema de Aquino et al. (2018a) é a representação das janelas de disponibilidade nas equipes. No LTPMOSP, cada equipe tem o tempo de disponibilidade de trabalho limitado a um único intervalo de tempo. No problema estudado nesta tese, esse intervalo é dividido em duas janelas, separadas por um período de indisponibilidade da equipe. Esse tempo de indisponibilidade entre as janelas pode representar, por exemplo, um período de férias ou algum outro tipo de folga acordado com a empresa.

A incorporação de manutenções com multi-habilidades e janelas de disponibilidade de equipe ao problema apresentado por Aquino et al. (2018a) dá origem ao LTPMOSP-MS. Até onde sabemos, essa extensão do problema ainda não foi estudada. Além disso, uma característica importante desse problema é que ele é NP-difícil, já que o LTPMOSP, uma versão específica do problema, também é NP-difícil, de acordo com Aquino et al. (2018a).

Além de introduzir um novo problema, este estudo pretende apresentar métodos de solução efetivos para encontrar boas soluções para o problema e capazes de gerar

bons limites superiores e inferiores. Para essa finalidade, propõe-se uma nova formulação de programação linear inteira (PLI) para o LTPMOSP-MS. A formulação proposta baseia-se em variáveis indexadas no tempo. Formulações deste tipo foram amplamente utilizadas para solucionar vários problemas de sequenciamento e são conhecidas na literatura (Unlu & Mason, 2010) por gerar limites inferiores fortes (no caso de problemas de minimização).

Como o problema é NP-difícil, esta tese também apresenta um algoritmo matheurístico de duas fases (denominado H2F) para resolvê-lo. De acordo com Ngoo et al. (2024), uma matheurística é um procedimento híbrido que combina heurísticas e métodos exatos. A primeira fase da H2F consiste em atribuir as manutenções às equipes, enquanto a segunda fase é responsável por programar as manutenções nas equipes e nas máquinas, respeitando as restrições do problema e as atribuições feitas na primeira fase. A atribuição na primeira fase é definida por uma nova formulação matemática e a programação da segunda fase é gerada por uma versão modificada da formulação PLI para o LTPMOSP-MS. São apresentadas três versões diferentes da heurística, sendo uma delas iterativa. Ambas as fases utilizam formulações baseadas em variáveis indexadas no tempo. No entanto, elas utilizam estratégias distintas para dividir o horizonte de planejamento. Como, na primeira fase, é solucionada a versão relaxada do LTPMOSP-MS, a solução ótima da formulação é um limite inferior para o problema. Na segunda fase, o LTPMOSP-MS é solucionado com todas as restrições, o que fornece um limite superior. Portanto, apesar da matheurística proposta nesta não ser um algoritmo exato, ela é capaz de provar a otimalidade das soluções ótimas encontradas se os limites inferior e superior convergirem.

1.2 Objetivos

O objetivo geral desta tese é desenvolver métodos eficientes, tanto exatos quanto heurísticos, para o LTPMOSP e o LTPMOSP-MS.

1.2.1 Objetivos específicos

Os objetivos específicos desta tese são:

(a) Realizar uma revisão ampla da literatura sobre trabalhos que investigam problemas com características semelhantes aos LTPMOSP e LTPMOSP-MS. Além disso, revisar e estudar as formulações matemáticas da literatura sobre o problema de sequenciamento de máquinas. Por fim, revisar e estudar algoritmos meta-heurísticos da literatura;

(b) Desenvolver novas formulações baseadas na literatura, bem como elaborar novas formulações utilizando estratégias novas, que sejam capazes de gerar limites inferiores mais fortes e mais efetivas em encontrar soluções exatas;

(c) Desenvolver um novo algoritmo para o LTPMOSP baseado nas meta-heurísticas estudadas na literatura que consiga gerar melhores planejamentos;

(d) Desenvolver uma matheurística que seja capaz de gerar boas soluções para o LTPMOSP-MS;

(e) Para o LTPMOSP, analisar o desempenho dos métodos de solução desenvolvidos utilizando as instâncias fornecidas por Aquino et al. (2018a). Além disso, pretende-se comparar a formulação proposta pela literatura com as propostas nesta pesquisa, quanto aos limites inferiores obtidos pela relaxação linear, ao tempo necessário para solucionar a formulação relaxada e ao número de soluções ótimas encontradas. Por fim, comparar os melhores limites superiores da literatura com os fornecidos pelo algoritmo heurístico desenvolvido nesta pesquisa;

(f) Para o LTPMOSP-MS, elaborar um amplo conjunto de instâncias. Analisar o desempenho dos métodos de solução desenvolvidos para esse conjunto de instâncias. Além disso, comparar os resultados computacionais da formulação com as diferentes versões da heurística matemática proposta.

1.3 Contribuições

A principal contribuição desta tese é a extensão do LTPMOSP, introduzido inicialmente por Aquino et al. (2018a). Esta extensão se apresenta em duas linhas de trabalho, descritas a seguir. A primeira linha de contribuições é associada ao próprio LTPMOSP:

(i) Proposição de duas novas formulações matemáticas para o LTPMOSP. A primeira formulação é baseada em variáveis indexadas no tempo e a segunda formulação é baseada em uma modelagem que combina o uso de variáveis indexadas no tempo com variáveis de precedência. Os resultados dos experimentos computacionais mostram que o desempenho das duas formulações se mostrou melhor que o desempenho da formulação proposta por Aquino et al. (2018a), em termos de tempo médio de resolução da formulação pelo solver, quantidade de instâncias resolvidas na otimalidade e *gap* médio, na resolução de instâncias de pequeno porte. Além disso, a segunda formulação proposta apresentou o melhor desempenho na resolução de instâncias de médio porte. Com esta formulação,

foi possível obter novas soluções exatas para algumas instâncias com até 600 ordens de manutenção.

(ii) Para a resolução de instâncias grandes do problema de forma eficiente, foi proposto um algoritmo baseado na meta-heurística GRASP , adaptado para o LTPMOSP, que utiliza um novo algoritmo de posicionamento de ordens de manutenção. Este algoritmo foi capaz de encontrar novos limites superiores para instâncias reais de larga escala, em um tempo significativamente menor do que o dos melhores resultados da literatura.

A segunda linha de contribuições se referem à introdução do LTPMOSP-MS como uma extensão do LTPMOSP:

(i) Proposição de uma formulação matemática, baseada em variáveis indexadas no tempo, para o LTPMOSP. Essa formulação utiliza a versão clássica da literatura sobre problemas de sequenciamento de variáveis indexadas no tempo. Os resultados dos experimentos computacionais mostram que, para um conjunto de instâncias com 20 a 80 manutenções, a formulação conseguiu encontrar solução ótima para aproximadamente 82% das instâncias. Para um conjunto de instâncias com 150 ou mais manutenções, a formulação não conseguiu gerar nenhuma solução.

(ii) Proposição de três versões diferentes de uma nova heurística de duas fases. A primeira fase do algoritmo consiste em atribuir manutenções às equipes, enquanto a segunda fase programa as manutenções nas equipes respeitando todas as restrições do LTPMOSP-MS. A solução da primeira fase é dada por uma formulação matemática, que utiliza uma estratégia nova na literatura para dividir o horizonte de planejamento. A solução dessa fase permite dividir o LTPMOSP-MS em um ou mais subproblemas, que são resolvidos na segunda fase. Uma versão da heurística soluciona vários subproblemas de forma independente e, em seguida, outra versão resolve um único subproblema na segunda fase. A terceira versão é iterativa e utiliza um pseudo-corte. Os experimentos computacionais mostram que o algoritmo H2F conseguiu encontrar limites inferiores e superiores melhores do que a formulação proposta para instâncias com até 80 manutenções. Para instâncias com 150 e 300 manutenções, o algoritmo H2F na versão iterativa conseguiu encontrar solução ótima para 50% das instâncias.

1.4 Organização do trabalho

Esta tese está organizada da seguinte maneira: o Capitulo 2 revisa alguns conceitos e formulações clássicas do problema de sequenciamento da produção, apresenta uma revisão da literatura relacionada aos problemas estudados e uma revisão de um método de solução utilizado na presente pesquisa. O Capítulo 3 apresenta uma descrição formal do LTPMOSP e a formulação matemática desenvolvida por Aquino et al. (2018a). No mesmo capítulo, são apresentadas as duas novas formulações matemáticas desenvolvidas e o algoritmo GRASP implementado para o LTPMOSP. Além disso, são exibidos os resultados dos experimentos computacionais realizados, bem como sua análise e a comparação com os resultados da literatura. Por fim, apresenta-se uma conclusão dos resultados. O Capítulo 4 apresenta uma descrição formal do LTPMOSP-MS, acompanhada de um exemplo didático do problema e de uma nova formulação matemática. No mesmo capítulo são explicadas as diferentes versões da H2F desenvolvidas para o LTPMOSP-MS. Além disso, ele apresenta, analisa e compara os resultados dos experimentos computacionais obtidos pelos métodos de solução propostos. Por fim, apresenta-se uma conclusão dos resultados. Por último, no Capítulo 5, são apresentadas as considerações finais e apresentadas as propostas de trabalhos futuros.

Capítulo 2

Revisão de Literatura

Este capítulo é dedicado à fundamentação teórica e revisão de trabalhos da literatura que possuem relação com a presente tese. A Seção 2.1 apresenta brevemente conceitos e definições relacionados a problemas de sequenciamento de produção. Além disso, ela apresenta uma revisão dos trabalhos que abordam problemas de sequenciamento de máquinas com características relacionadas ao LTPMOSP e formulações matemáticas para problemas de sequenciamento da produção baseadas em diferentes tipos de variáveis de decisão encontrados na literatura associada. A Seção 2.2 revisa trabalhos da otimização de planejamento de manutenções preventivas que possuem relação com o LTPMOSP. Além disso, faz um resumo dos trabalhos revisados, destacando a importância da pesquisa feita nessa tese em relação ao estado da arte para problemas com características semelhantes ao LTPMOSP. A Seção 2.3 revisa estudos que abordam problemas de sequenciamento com multi-habilidade. A Seção 2.4, por sua vez, faz uma breve revisão de literatura de trabalhos que envolvem planejamento de manutenções preventivas com multi-habilidade. Além disso, destaca que o LTPMOSP-MS é um problema novo na literatura. A Seção 2.5.1 apresenta a metaheurística GRASP, que é utilizada como método de solução base desta proposta de tese. Por fim, a Seção 2.6 faz as considerações finais relacionadas a esse capítulo.

2.1 Problemas de Sequenciamento de Produção

Os problemas de sequenciamento têm recebido muita atenção dos pesquisadores nos últimos 50 anos (Agnetis et al., 2025). Essas pesquisas têm sido motivadas por questões que surgem no planejamento de produção, telecomunicações, logística, controle de computadores, para citar apenas algumas aplicações. A recente revisão de Agnetis et al. (2025), presenta uma visão geral das pesquisas realizadas ao longo da última metade do século sobre sequenciamento. O foco principal dessa revisão

são pesquisas que envolvem o sequenciamento de máquinas, que é exatamente o foco desta seção.

Para Agnetis et al. (2025), um problema de sequenciamento consiste em trabalhar com alocação de recursos limitados a atividades ao longo do tempo, visando a otimização de um ou vários objetivos. Os recursos em uma organização podem assumir diferentes formas, como máquinas em um centro de trabalho, pistas dos aeroportos, equipes em um canteiro de obras, unidades de processamento em um ambiente de computação, etc. As atividades podem ser encomendas a serem distribuídas para clientes, operações em um processo de produção, decolagens e aterrissagens de aviões num aeroporto, execuções de programas de computador, dentre outras. Essas atividades podem ter uma ou mais características, como tempo de processamento de cada recurso, data limite de inicialização e finalização, importância, entre outras. Os objetivos a serem otimizados também variam, sendo os mais comuns os baseados nos tempos de conclusão e nas datas de entregas.

Em uma abordagem formal do problema, seja $J = \{1, ..., n\}$ o conjunto de tarefas que devem ser processadas em um conjunto $M = \{1, ..., m\}$ de máquinas. Cada tarefa j está associado a um tempo de processamento p_j necessário para a execução da tarefa j; p_{jk}, caso o tempo de processamento esteja associado à máquina k. Além disso, uma tarefa j também pode estar associada a:

(i) data de liberação r_j, na qual a tarefa j pode começar a ser processada, não sendo permitido o seu início antes deste instante;

(ii) data de entrega d_j, na qual a tarefa j deve, idealmente, ser finalizada. Caso o processamento termine após a data de entrega, uma penalidade pode ser considerada. Quando a data deve ser atendida é referido como *deadline* e denotado por $\overline{d}_j$;

(iii) peso w_j que indica a importância relativa da tarefa j em relação às demais.

As várias características de tarefas, máquinas e sequenciamento podem gerar inúmeras variantes de um problema de sequenciamento. Com o objetivo de representar de maneira simplificada cada variante, Graham et al. (1979) propuseram a classificação de três campos, denotada como $\alpha|\beta|\gamma$. O campo α descreve o ambiente da máquina e contém apenas uma entrada (para ver as possibilidades de entradas, veja a Tabela 2.1). O campo β provê detalhes das características do processamento e das restrições, podendo conter uma ou múltiplas entradas, ou mesmo ser vazio (veja Tabela 2.2). O campo γ denota a função objetivo do problema, ou critério de otimalidade (veja Tabela 2.3). As funções objetivo podem estar associadas à variável C_j que representa o tempo de conclusão da tarefa j.

Tabela 2.1: Entradas para o campo α.

α	**Ambiente**	**Definição**
1	Máquina única	Todas as tarefas podem ser processadas em apenas uma máquina.
P_m	Máquinas paralelas idênticas	Existem m máquinas paralelas idênticas disponíveis para processar as n tarefas, com tempos de processamento iguais em todas as máquinas.
Q_m	Máquinas paralelas uniformes	Existem m máquinas em paralelo, porém, com diferentes fatores de velocidade v_k para cada máquina k. Assim, o tempo de processamento da tarefa j na máquina k é dado por $p_{jk} = v_k \times p_j$.
R_m	Máquinas não relacionadas	Existem m máquinas em paralelo. A máquina k processa a tarefa j com a velocidade v_{jk}. Ou seja, a velocidade da máquina depende da tarefa que está sendo processada.
O	*Open shop*	As tarefas devem ser processadas em todas as máquinas em p_{jk} unidades de tempo. Não há restrição quanto à sequência de execução das tarefas.
J	*Job shop*	Cada tarefa j possui uma rota predefinida a ser seguida, geralmente distinta das demais.
F	*Flow shop*	Existem $\|M\|$ máquinas em série. Cada tarefa j deve ser processada por cada uma das máquinas disponíveis. Todas as tarefas devem seguir a mesma rota.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2022)

Tabela 2.2: Entradas para o campo β.

β	**Termo**	**Definição**
pmtn	preempção	Preempção implica que uma tarefa pode ser dividida em várias etapas, ou seja, seu processamento pode ser interrompido e retomado posteriormente na mesma ou em outra máquina.
prec	precedência	As restrições de precedência podem ocorrer em ambientes de máquina única ou paralelos, exigindo que uma ou mais tarefas sejam concluídas antes que outra possa iniciar seu processamento.

S_j	tempos de *setup*	Quando s_j está presente, cada tarefa j possui um tempo de preparação da máquina, S_j, para iniciar seu processamento. Se esse tempo depender da sequência das tarefas, então é adicionado o índice i, ou seja, S_{ij}, referente a tarefa i que precede a tarefa j. Além disso, se o tempo de *setup* entre as tarefas i e j depende da máquina k, é adicionado um terceiro índice k, ou seja, s_{ijk}.
M_j	elegibilidade de máquinas	Quando M_j está presente, nem todas as m máquinas são capazes de processar a tarefa j. O conjunto M_j denota o conjunto de máquinas que podem processar a tarefa j. O símbolo M_j aparece em ambientes com múltiplas máquinas.
brkdwn	*breakdowns*	As quebras de máquinas implicam que uma máquina pode não estar continuamente disponível. Os períodos em que uma máquina não está disponível são assumidos como fixos.
$batch(b)$	*batch*	A máquina pode processar simultaneamente um número de tarefas (seja b esse número). Os tempos de processamento das tarefas em um *batch* podem não ser todos iguais e o *batch* inteiro é terminado somente quando o último trabalho do *batch* for concluído, implicando que o tempo de conclusão do lote inteiro é determinado pelo trabalho com o maior tempo de processamento.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2022)

Tabela 2.3: Entradas para o campo γ.

γ	**Definição**
C_{max}	O *makespan*, definido como $max(C_1, ..., C_n)$, é equivalente ao tempo de conclusão da última tarefa a ser processada pelo sistema.
$\sum C_j$	A soma dos tempos de conclusão das n tarefas disponíveis.
$\sum w_j C_j$	A soma ponderada dos tempos de conclusão das n tarefas. Neste caso de função objetivo, cada tarefa j possui um peso w_j associado.
L_{max}	O atraso máximo é definido como $\max(L_1, ..., L_n)$. O atraso L_j da tarefa j é obtido pela diferença entre o tempo real e o tempo desejado da sua conclusão.
$\sum w_j T_j$	A soma ponderada dos atrasos das n tarefas disponíveis. $T_j = max\{0, L_j\}$

$\sum w_j U_j$	A soma ponderada do número de tarefas atrasadas.

Fonte: Adaptado de Graham et al. (1979) e Pinedo (2022).

Uma das principais linhas de pesquisa em sequenciamento de produção é determinar a complexidade do problema. Em Brucker et al. (2002); Brucker (2004); Brucker & Knust (2009) e Pinedo (2022) são apresentadas análises de complexidade para variantes do problema de sequenciamento de produção. Os resultados dos trabalhos citados e da literatura indicam que a maioria dos problemas de sequenciamento é classificada como $\mathcal{NP}$-difíceis, principalmente aqueles em um ambiente de máquinas paralelas.

2.1.1 Literatura relacionada

Na extensa literatura sobre problemas de sequenciamento de máquinas, a maioria dos trabalhos considera problemas com função objetivo relacionada ao tempo de processamento final das tarefas. Todos os critérios de otimalidade da Tabela 2.3 estão associados ao tempo de conclusão das tarefas. Como o tempo final é definido pelo sequenciamento das tarefas, todas são processadas. Diferentemente desses trabalhos, baseado na analogia feita na Seção 1.1, um dos objetivos do LTPMOSP é alocar o máximo de tarefas no mínimo de máquinas (no caso, equipes de trabalho). Consequentemente, não se garante que todas as tarefas sejam alocadas e executadas. A seguir, são revisados trabalhos que estudam problemas de sequenciamento de máquinas com características semelhantes às do problema abordado neste trabalho.

Rojanasoonthon & Bard (2005) estudam um problema de sequenciamento de máquinas paralelas não relacionadas, com janelas de tempo ($R_m|r_j, \overline{d}_j, M_j, s_{jk}|\sum w_j U_j$), no qual as tarefas podem ser processadas apenas dentro de sua respectiva janela. Além disso, as tarefas possuem diferentes classes de prioridade, e isso é considerado na função objetivo por meio do parâmetro de peso w_j. O objetivo do problema é maximizar o número ponderado de tarefas processadas. Para solucionar o problema, propõe-se uma heurística baseada em programação dinâmica e em GRASP. Rojanasoonthon & Bard (2006) estudam um problema semelhante ao de Rojanasoonthon & Bard (2005), que também buscam maximizar o número de tarefas processadas, mas com a diferença de que as tarefas possuem múltiplas janelas. Com essa diferença, surge um novo problema e uma nova abordagem de solução. Os autores do artigo citado desenvolvem um algoritmo *branch-and-price* para resolver instâncias do problema.

Gholami et al. (2019) focam em um problema de sequenciamento de máquinas paralelas uniformes ($Q_m|r_j, d_j|\gamma$) com o objetivo de maximizar a receita e o número

de tarefas. Como abordagem de solução, propõe-se uma heurística de duas fases, denominada *Neighborhood combined dispatching rules algorithm*. Esta heurística é comparada com a meta-heurística *Simulated Annealing*, e apresenta um melhor desempenho. Os autores citam que os trabalhos que estudam objetivos como maximizar a receita e o número de tarefas processadas são limitados.

Em uma linha de pesquisa de problemas que não assumem que todas as tarefas devem ser processadas, o *Order Acceptance and Scheduling Problem* (OASP) busca definir quais tarefas são aceitas para o processamento e determinar uma programação para elas, com o objetivo de maximizar a receita. Para uma revisão da literatura sobre o OASP, veja Slotnick (2011). Os esforços dos pesquisadores do OASP têm se voltado principalmente para os problemas com máquina única (Slotnick & Morton, 2007; Oğuz et al., 2010; Nobibon & Leus, 2011; Cesaret et al., 2012; Lin & Ying, 2013; Geramipour et al., 2017; Chaurasia & Singh, 2017; Silva et al., 2018; Tarhan & Oğuz, 2021, 2022). Uma das variantes do OASP que teve um maior número de pesquisas é o problema $1|r_j, d_j, \overline{d}_j, s_{ij}|\gamma$, em que foram propostas diferentes abordagens de solução. Este problema tem como objetivo maximizar a receita total, dada pela diferença entre a soma das receitas das tarefas processadas e a soma ponderada dos atrasos. Enquanto os problemas com máquina única receberam maior foco dos pesquisadores, os problemas com máquinas paralelas (Wang et al., 2015; Emami et al., 2016, 2017; Ou & Zhong, 2017; Wu et al., 2018; Wang & Ye, 2019; Liu & Lu, 2020; Naderi & Roshanaei, 2020) receberam menor atenção. Em Wang et al. (2015); Wu et al. (2018) e Naderi & Roshanaei (2020) são estudados problemas de sequenciamento de máquinas paralelas idênticas com o objetivo de maximizar a receita líquida total, definida como a diferença entre a soma das receitas das ordens aceitas e o atraso total ponderado. Wang & Ye (2019) focam em um problema com a mesma função objetivo, porém em um ambiente com máquinas paralelas não relacionadas. Emami et al. (2016, 2017) também consideram um problema com a mesma função objetivo dos artigos citados, porém em um ambiente com máquinas paralelas uniformes e uma abordagem exata e robusta do problema, considerando incerteza poliedral na receita das tarefas executadas e no tempo de processamento. Ou & Zhong (2017) e Liu & Lu (2020) são estudos dedicados a problemas com máquinas idênticas e com o objetivo de minimizar o *makespan* de todas as tarefas aceitas mais a soma total das penalidades das tarefas rejeitadas.

Um outro problema da literatura que tem uma característica semelhante ao abordado neste trabalho é estudado em Rebai et al. (2012). O artigo citado estuda um problema de máquina única para sequenciar manutenções preventivas, com o objetivo de minimizar a antecipação e o atraso das manutenções. Neste problema, são consideradas janelas de tempo em que a manutenção deve ser executada, para evitar custos

de antecipação e atraso, contabilizados na função objetivo. Ainda sobre o problema, todas as manutenções devem ser executadas, diferentemente do LTPMOSP.

2.1.2 Formulações matemáticas

Embora a literatura sobre os problemas de sequenciamento apresente uma grande variedade de abordagens de solução heurísticas, há um grande esforço de pesquisadores voltado ao desenvolvimento de formulações matemáticas. Essas formulações são métodos comprovados, capazes de gerar limitantes para diversos problemas de sequenciamento (Unlu & Mason, 2010). Esses limitantes podem levar ao desenvolvimento de abordagens de solução, denominadas na literatura como exatas. Através dela é possível obter soluções ótimas, as quais são as melhores soluções factíveis para o problema. Essa abordagem é a solução da formulação matemática via algoritmos exatos ou *solver* comerciais (Unlu & Mason, 2010). Esta pesquisa estuda a resolução de instâncias do problema através da solução de formulações matemáticas via *solver* comercial.

As formulações encontradas na literatura para lidar com os problemas de sequenciamento de máquinas podem ser classificadas conforme as variáveis de decisão em que se baseiam (Unlu & Mason, 2010). Existem vários tipos diferentes de variáveis de decisão com os quais os pesquisadores já modelaram e otimizaram problemas de sequenciamento de máquinas. O artigo de Unlu & Mason (2010) avalia formulações baseadas em quatro tipos de variáveis para solucionar variantes do problema de sequenciamento de máquinas paralelas não preemptivas. A seguir, são revisados diversos tipos de variáveis encontradas na literatura. Para cada tipo de variável, foi elaborada uma formulação para o problema de máquinas paralelas idênticas, com o objetivo de minimizar a soma dos tempos de conclusão das tarefas processadas.

2.1.2.1 Formulação baseada em variáveis de tempo de conclusão

As variáveis de tempo de conclusão foram introduzidas por Manne (1960) em um problema de sequenciamento de *job shop*. Alguns anos depois, Balas (1985) utilizou este tipo de variável em sua conhecida formulação disjuntiva de *job shop*. Posteriormente, a formulação inicial de Balas (1985) também foi estudada por Queyranne & Wang (1991) e Queyranne (1993).

Nas formulações baseadas em variáveis de tempo de conclusão, a sequência é determinada exclusivamente pelos tempos de conclusão das tarefas respectivas. Ela utiliza o parâmetro B, que possui um valor constante e muito grande e as variáveis de decisão: $x_j^m = 1$ se a tarefa j é processada na máquina m e 0, caso contrário; $y_{ij}^m = 1$ se as tarefas i e j são processadas na mesma máquina m, em que i precede j; e C_j

representa o tempo de conclusão da tarefa j. Além disso, esta formulação considera a tarefa zero, cujo tempo de processamento é zero. Essa tarefa deve ser a primeira na sequência de qualquer máquina. A formulação baseada nas variáveis de tempo de conclusão para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{j\in J} C_j \tag{2.1}$$

$$\textbf{suj. a:} \sum_{m\in M} x_j^m = 1 \qquad \forall j \in J \tag{2.2}$$

$$y_{ij}^m + y_{ji}^m \geq x_j^m + x_i^m - 1 \qquad \forall m \in M, i \in (J \cup 0), j \in J, i \neq j \tag{2.3}$$

$$C_j \geq C_i + \sum_{m\in M} p_j y_{ij}^m - (1 - \sum_{m\in M} y_{ij}^m)B \qquad \forall i \in (J \cup 0), j \in J, i \neq j \tag{2.4}$$

$$y_{0j}^m = x_j^m \qquad \forall m \in M, j \in J \tag{2.5}$$

$$C_j \geq 0 \qquad \forall j \in (J \cup 0) \tag{2.6}$$

$$x_j^m \in \{0,1\} \qquad \forall m \in M, j \in (J \cup 0) \tag{2.7}$$

$$y_{ij}^m \in \{0,1\} \qquad \forall m \in M, i \in (J \cup 0), j \in J, i \neq j \tag{2.8}$$

A função objetivo (2.1) minimiza a soma dos tempos de conclusão. As restrições (2.2) impõem que cada tarefa j seja atribuída a uma máquina m. As restrições (2.3) definem a precedência das tarefas i e j. As restrições (2.4) definem o tempo de conclusão da tarefa j, baseado no tempo de conclusão da tarefa i que a precede. As restrições (2.5) garantem que, se a tarefa j é alocada à máquina m, então a tarefa zero precede as demais tarefas executadas na máquina. As restrições (2.6)-(**??**

As variáveis de ordenação linear foram introduzidas por Potts (1980) para resolver um problema de máquina única com restrições de precedência. Elas são denominadas por Pinedo (2022) como variáveis de sequenciamento. Dyer & Wolsey (1990) usaram essas variáveis, em combinação com variáveis de tempo de início, para estudar o problema de máquina única com restrições de data de liberação. Além disso, Dyer & Wolsey (1990) desenvolveram várias abordagens poliédricas. As variáveis de ordenação linear foram estudadas também por Blazewicz et al. (1991), Nemhauser & Savelsbergh (1992) e Chudak & Hochbaum (1999).

A formulação desta subseção se baseia no uso de três variáveis binárias para ordenar as tarefas nas máquinas. Estas variáveis são: $y_{ij} = 1$ se a tarefa i precede j e 0, caso contrário. Note que a tarefa i não é necessariamente posicionada imediatamente antes da tarefa j quando $y_{ij} = 1$; $x_j^m = 1$ se a tarefa i é posicionada na máquina m, e 0, caso contrário; e $v_{ij} = 1$ se as tarefas i e j não são alocadas na mesma máquina. Além disso, a formulação utiliza a variável C_j^m, que representa o tempo de conclusão da tarefa j na máquina m. A formulação baseada nas variáveis de ordenação linear para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{m\in M}\sum_{j\in J} C_j^m \tag{2.9}$$

$$\textbf{suj. a:} \sum_{m \in M} x_j^m = 1 \qquad \forall j \in J \quad (2.10)$$

$$y_{ij} + y_{ji} + v_{ij} = 1 \qquad \forall i, j \in J, i \neq j \quad (2.11)$$

$$y_{ij} + y_{jk} + y_{ki} \leq 2 \qquad \forall i, j, k \in J, i \neq j \neq k \quad (2.12)$$

$$x_i^m + x_j^m + v_{ij} \leq 2 \qquad \forall m \in M, i, j \in J, i \neq j \quad (2.13)$$

$$C_j^m \geq p_j x_j^m \qquad \forall m \in M, j \in J \quad (2.14)$$

$$C_j^m \geq C_i^m - (1 - y_{ij})B + p_j(y_{ij} + x_i^m + x_j^m - 2) \qquad \forall m \in M, i, j \in J, i \neq j \quad (2.15)$$

$$C_j^m \geq 0 \qquad \forall m \in M, j \in J \quad (2.16)$$

$$x_j^m \in \{0, 1\} \qquad \forall m \in M, j \in J \quad (2.17)$$

$$y_{ij}, \delta_{ij} \in \{0, 1\} \qquad \forall i, j \in J, i \neq j \quad (2.18)$$

A função objetivo (2.9) minimiza a soma dos tempos de conclusão. As restrições (2.10) impõem que cada tarefa j seja atribuída a uma máquina m. As restrições (2.11) garantem a precedência das tarefas i e j, desde que ambas estejam alocadas na mesma máquina. As restrições (2.12) são responsáveis pela ordem linear de três tarefas. As restrições (2.13) garantem que a variável v_{ij} será igual a zero se as tarefas i e j forem alocadas à mesma máquina. As restrições (2.14) garantem que o tempo de conclusão da tarefa é maior ou igual ao tempo de processamento da mesma, quando atribuída à máquina m. As restrições (2.15) determinam o tempo de conclusão das tarefas. As restrições (2.16)-(2.18) definem o domínio das variáveis de decisão.

2.1.2.2 Formulação baseada em variáveis de atribuição e posicionamento

As variáveis de atribuição e de posicionamento foram inicialmente aplicadas a problemas de sequenciamento de máquina por Lasserre & Queyranne (1992). Os autores estudam um problema de máquina única como um sistema que pode ser controlado em instantes discretos no tempo, por meio de uma combinação de controles discretos e contínuos. Além disso, o problema de minimizar o número total de tarefas atrasadas em uma única máquina foi estudado por Dauzere-Peres (1997) e Dauzère-Pérès & Sevaux (2003) usando variáveis de atribuição e de datas de posicionamento.

Em uma formulação com variáveis de atribuição e posicionamento, as variáveis de decisão são definidas com base na noção de que cada máquina possui um número fixo de posições, ou *slots*, às quais as tarefas podem ser atribuídas. Essas posições especificam a posição relativa de uma tarefa em relação às demais processadas na mesma máquina, gerando, assim, uma sequência de tarefas na máquina. Esta formulação utiliza as variáveis de decisão: $x_{jl}^m = 1$ se a tarefa j é atribuída à posição l na máquina m e 0, caso contrário; Δ_l^m que denota o tempo de conclusão da tarefa na posição l na máquina m; e C_j que denota o tempo de conclusão da tarefa j. Seja $l \in L = \{1, ..., |J|\}$, em que $|J|$ é o número de tarefas disponíveis e B é uma constante com valor muito grande. A formulação baseada em variáveis de atribuição e

posicionamento para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{j\in J} C_j \tag{2.19}$$

$$\textbf{suj. a:} \sum_{m\in M}\sum_{l\in L} x_{jl}^m = 1 \qquad \forall j \in J \tag{2.20}$$

$$\sum_{j\in J} x_{jl}^m \leq 1 \qquad \forall l \in L, m \in M \tag{2.21}$$

$$\Delta_1^m \geq p_j x_{j1}^m \qquad \forall m \in M \tag{2.22}$$

$$\Delta_l^m \geq \Delta_{l-1}^m + \sum_{j\in J} p_j x_{jl}^m \qquad \forall m \in M, l \in L : l > 1 \tag{2.23}$$

$$C_j \geq \Delta_l^m - B(1 - x_{jl}^m) \qquad \forall m \in M, j \in J, l \in L \tag{2.24}$$

$$\Delta_l^m \geq 0 \qquad \forall j \in J, l \in L \tag{2.25}$$

$$C_j \geq 0 \qquad \forall j \in J \tag{2.26}$$

$$x_{jl}^m \in \{0, 1\} \qquad \forall m \in M, j \in J, l \in L \tag{2.27}$$

A função objetivo (2.19) minimiza a soma dos tempos de conclusão. As restrições (2.20) garantem que todas as tarefas são atribuídas a exatamente uma posição e a uma única máquina. As restrições (2.21) garantem que cada posição em todas as máquinas contém no máximo uma tarefa. As restrições (2.22) garantem que o tempo de conclusão da tarefa alocada na primeira posição de cada máquina seja maior ou igual ao tempo de processamento da respectiva tarefa. As restrições (2.23) determinam o tempo de conclusão das tarefas em cada posição de cada máquina. As restrições (2.24) determinam o tempo de conclusão de cada tarefa, independentemente de sua posição. As restrições (2.25)-(2.27) definem o domínio das variáveis de decisão.

2.1.2.3 Formulação baseada em variáveis de rede

As variáveis de rede foram usadas inicialmente para modelar o problema de sequenciamento de uma única máquina com tempo de processamento com dependência de sequência em Queyranne & Schulz (1994), devido a sua proximidade com o problema de caixeiro viajante com dependência de tempo (Fox et al., 1980). Cakici & Mason (2007) apresentaram uma formulação baseada em rede para o problema de sequenciamento em máquinas paralelas com restrições de recursos auxiliares. Essa formulação difere das demais porque as restrições são formuladas com base na ideia de redes.

Nos modelos com estas variáveis para problemas de máquinas paralelas, as tarefas são representadas por nós da rede, e o planejamento de uma máquina é representado por um caminho na rede. A sequência de nós visitados nesse caminho representa a sequência na qual as tarefas são processadas por uma máquina. Neste tipo de formulação, cada caminho é iniciado com a tarefa fictícia zero, que possui o tempo de processamento igual a zero. A formulação utiliza as variáveis de decisão: $x_{ij}^m = 1$ se a tarefa i precede imediatamente a

tarefa j na máquina m e 0, caso contrário; e C_j que denota o tempo de conclusão da tarefa j. Esta formulação para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{j\in J} C_j \tag{2.28}$$

$$\textbf{suj. a:} \sum_{m\in M}\sum_{i\in J\cup\{0\}:i\neq j} x_{ij}^m = 1 \qquad \forall j \in J \tag{2.29}$$

$$\sum_{j\in J} x_{0j}^m \leq 1 \qquad \forall m \in M \tag{2.30}$$

$$\sum_{j\in J:i\neq j} x_{ij}^m \leq 1 \qquad \forall m \in M, i \in J \tag{2.31}$$

$$\sum_{i\in J\cup\{0\}:i\neq j} x_{ij}^m = \sum_{i\in J\cup\{|J|+1\}:j\neq i} x_{ji}^m \qquad \forall m \in M, j \in J \tag{2.32}$$

$$C_j \geq C_i - (1 - \sum_{m\in M} x_{ij}^m)B + \sum_{m\in M} p_j x_{ij}^m \qquad \forall i \in J\cup\{0\}, j \in J : i \neq j \tag{2.33}$$

$$C_j \geq 0 \qquad \forall j \in J \tag{2.34}$$

$$x_{ij}^m \in \{0,1\} \qquad \begin{array}{r}\forall m \in M,\\ i \in J\cup\{0\},\\ j \in J\cup\{|J|+1\} : i \neq j\end{array} \tag{2.35}$$

A função objetivo (2.28) minimiza a soma dos tempos de conclusão. As restrições (2.29) garantem que toda tarefa é alocada a uma máquina que possui apenas um único antecessor. Os conjuntos de restrições (2.30) garantem que, no máximo, uma tarefa seja programada imediatamente após o trabalho fictício zero. A variável $x_{0j}^m = 1$ indica que a tarefa j é a primeira a ser processada na máquina m. As restrições (2.31) garantem que, em cada máquina, a tarefa i terá, no máximo, um sucessor. As restrições (2.32) garantem a sequência das tarefas nas máquinas. As restrições (2.33) determinam os tempos de conclusão das tarefas. As restrições (2.34) e (2.35) definem o domínio das variáveis de decisão.

2.1.2.4 Formulação baseada em variáveis indexadas no tempo

A formulação baseada em variáveis indexadas no tempo para problemas de sequenciamento de máquinas foi introduzida por Sousa & Wolsey (1992) com aplicação em um problema com uma única máquina.

Akker et al. (2000) e Bigras et al. (2008) utilizaram formulações com variáveis indexadas no tempo e propuseram a aplicação do método de geração de colunas para resolver o problema. Berghman, Lotte et al. (2021) abordaram o problema de sequenciamento de máquinas paralelas não relacionadas por meio de uma formulação baseada em variáveis indexadas no tempo. Os autores apresentam novas inequações válidas e um algoritmo *cutting-planes* para resolver instâncias do problema.

Nas formulações indexadas no tempo, o horizonte de planejamento é dividido em períodos com duração igual a δ. Em geral, as variáveis de decisão desta formulação definem o

período de início de processamento de cada tarefa. Após essa definição, é gerada a programação das tarefas em cada máquina, utilizando as seguintes variáveis de decisão: $x_{jm}^t = 1$ se a tarefa j for programada para começar seu processamento no período t na máquina m e 0, caso contrário; e C_j que denota o tempo de conclusão da tarefa j. Seja o conjunto de períodos $T = \{0, 1, ..., t_{\max}\}$, onde $t_{\max}$ é o limite superior dos tempos de conclusão das tarefas. A formulação baseada em variáveis indexadas no tempo para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{j \in J} C_j \tag{2.36}$$

$$\textbf{suj. a:} \sum_{m \in M} \sum_{t=0}^{t_{\max}-p_j+1} x_{jm}^t = 1 \qquad \forall j \in J \tag{2.37}$$

$$\sum_{j \in J} \sum_{h=\max(0,t-p_j+1)}^{t-1} x_{jm}^h \leq 1 \qquad \forall m \in M, t \in \{0, ..., t_{max} - p_j + 1\} \tag{2.38}$$

$$C_j \geq \sum_{m \in M} \sum_{t=0}^{t_{\max}-p_j+1} (t + p_j) x_{jm}^t \qquad \forall j \in J \tag{2.39}$$

$$C_j \geq 0 \qquad \forall j \in J \tag{2.40}$$

$$x_{jm}^t \in \{0, 1\} \qquad \forall m \in M, j \in J, t \in T \tag{2.41}$$

A função objetivo (2.36) minimiza a soma dos tempos de conclusão. As restrições (2.37) garantem que cada tarefa inicia o processamento em apenas uma máquina, em apenas um instante, ou seja, em apenas um período. As restrições (2.38) permitem que, no máximo, uma tarefa seja processada a qualquer momento em qualquer máquina. As restrições (2.39) determinam o tempo de conclusão de cada tarefa j. As restrições (2.40) e (2.41) definem o domínio das variáveis de decisão.

2.1.2.5 Formulação baseada em variáveis de arcos indexados no tempo

A formulação baseada em variáveis de arcos indexados no tempo foi introduzida por Pessoa et al. (2010) para estudar um problema de sequenciamento de máquinas paralelas idênticas que busca minimizar a soma ponderada dos atrasos. As formulações indexadas no tempo, revisadas na Subseção 2.1.2.4, são conhecidas por gerar bons limites inferiores (no caso de problemas de minimização) (Unlu & Mason, 2010). No entanto, a formulação proposta por Pessoa et al. (2010) consegue gerar limites inferiores ainda melhores. Além disso, os autores do artigo citado desenvolveram um algoritmo *branch-and-price* com base na nova formulação que propuseram. Estudando o mesmo tipo de variáveis, Silva et al. (2018) apresentam uma formulação que permite tempo ocioso entre as tarefas. Nesello et al. (2018) e Silva et al. (2021) também elaboram formulações baseadas em variáveis de arcos indexadas no tempo, que essa subseção foca.

A formulação baseada em variáveis de arcos indexadas no tempo também divide o hori-

zonte de planejamento em períodos, como na formulação baseada em variáveis indexadas no tempo discutida na Subseção 2.1.2.4. Ela utiliza a tarefa fictícia zero para indicar o início e $|J|+1$ para indicar o fim de uma sequência. Os tempos de processamento de ambas as tarefas são nulos. Esta formulação utiliza as variáveis de decisão: $x^t_{ijm} = 1$ se a tarefa j inicia o processamento em t na máquina m após a tarefa i ter sido finalizada, e 0, caso contrário; e C_j, que denota o tempo de conclusão da tarefa j. As variáveis binárias x^t_{ijm} são responsáveis por sequenciar as tarefas. Seja o conjunto $W = \{(i,j) : i \in J \cup \{0\}$ e $j \in J \cup \{|J|+1\}, i \neq j\}$. A formulação baseada em variáveis de arcos indexados no tempo para o problema $P_m||\sum C_j$ é, então:

$$\min \quad \sum_{j \in J} C_j \tag{2.42}$$

$$\textbf{suj. a:} \sum_{m \in M} \sum_{(i,j) \in W} \sum_{t=p_i}^{t_{\max}-p_j} x^t_{ijm} = 1 \qquad \forall j \in J \tag{2.43}$$

$$\sum_{(0,j) \in W} \sum_{t=p_0}^{t_{\max}-p_j} x^t_{0jm} \leq 1 \qquad \forall m \in M \tag{2.44}$$

$$\sum_{\substack{(i,j) \in W: \\ t \geq p_j}} x^t_{ijm} - \sum_{\substack{(j,i) \in W: \\ t+p_j \leq t_{\max}-p_i}} x^{t+p_j}_{jim} = 0 \qquad \forall m \in M, j \in J, \; t \in T : t \leq t_{max} - p_j \tag{2.45}$$

$$C_j \geq \sum_{m \in M} \sum_{(i,j) \in W} \sum_{t=p_i}^{t_{max}-p_j} (t+p_j) x^t_{ijm} \qquad \forall j \in J \tag{2.46}$$

$$C_j \geq 0 \qquad \forall j \in J \tag{2.47}$$

$$x^t_{ijm} \in \{0,1\} \qquad \forall m \in M, (i,j) \in W, \; t \in \{p_i, ..., (t_{\max} - p_j)\} \tag{2.48}$$

A função objetivo (2.42) minimiza a soma dos tempos de conclusão. As restrições (2.43) impõem que toda tarefa seja visitada por exatamente um caminho. As restrições (2.44) garantem que existirá no máximo uma tarefa iniciando a sequência de cada máquina. As restrições (2.45) garantem que, se a tarefa j é processada no período t, a tarefa i seguinte $(i \neq j)$ deve ser processada no período $t + p_j$. As restrições (2.46) determinam o tempo de conclusão de cada tarefa j. As restrições (2.47) e (2.48) definem o domínio das variáveis de decisão.

2.1.2.6 Comparação das dimensões das formulações

Os diferentes tipos de variáveis originam formulações com diferentes expressões e dimensões. A Tabela 2.4 apresenta o número de restrições e de variáveis para as formulações que resolvem o problema $P_m||\sum C_j$.

A segunda coluna da Tabela 2.4 mostra que os termos que mais influenciam o número de restrições nas formulações das Seções 2.1.2.1, 2.1.2.2, 2.1.2.3 são de segunda ordem.

Tabela 2.4: Dimensões das formulações do problema $P_m||\sum C_j$.

Seção da Formulação	# de restrições	# de variáveis
2.1.2.1	$\|J\|^2(\|M\|+1)+\|M\|.\|J\|+\|J\|$	$\|M\|.\|J\|^2+\|M\|.\|J\|+\|M\|+\|J\|+1$
2.1.2.1	$\|J\|^3+2\|M\|\|J\|^2-2.\|J\|^2-\|M\|.\|J\|+2\|J\|$	$2\|J\|^2+\|M\|.\|J\|$
2.1.2.2	$\|M\|.\|J\|^2+2\|M\|.\|J\|+\|J\|$	$\|J\|^2(\|M\|+1)+\|I\|$
2.1.2.3	$\|J\|^2+2\|M\|.\|J\|+\|M\|+\|J\|$	$\|M\|.\|J\|^2+\|M\|.\|J\|+2\|J\|$
2.1.2.4	$\|M\|.\|T\|+\|M\|+2\|J\|-\sum_{j\in J}p_j/\delta$	$\|M\|.\|J\|.\|T\|+\|J\|$
2.1.2.5	$\|M\|.\|J\|.\|T\|+2\|J\|+\|M\|-\sum_{j\in J}p_j/\delta$	$\|M\|.\|J\|^2.\|T\|+\|M\|.(\|J\|+1)\|T\|+\|J\|-\sum_{j\in J}p_j/\delta$

Apenas na formulação da Seção 2.1.2.1, esse termo é de terceira ordem. Diferente dessas formulações, as das Seções 2.1.2.4 e 2.1.2.5 dependem de um termo pseudo-polinomial. No entanto, esse termo depende diretamente do número de períodos, o qual pode ser um número relativamente grande para casos com horizontes de planejamento longos. Quando a mesma análise é feita na terceira coluna, verifica-se que, apenas na formulação da Seção 2.1.2.4, o termo que mais influencia o número de variáveis é o pseudo-polinomial. Nas demais formulações, ele é de segunda ordem. Porém, o número de períodos pode ser muito maior do que o de tarefas, dependendo do tamanho do horizonte de planejamento.

2.2 Otimização de planejamento de manutenções preventivas

A literatura da otimização de manutenção é extensa, com inúmeras linhas de pesquisa. Dentre essas linhas, estão os problemas de sequenciamento de manutenção preventiva, com aplicações em diversas áreas, utilizando vários tipos de abordagens. Algumas das principais áreas de aplicação são, por exemplo, o planejamento de manutenções em sistemas de energia elétrica (Froger et al., 2016) e o planejamento de manutenções em vias ferroviárias (Sedghi et al., 2021). Uma revisão de literatura sobre otimização de manutenções é apresentada por Jonge & Scarf (2020).

Por outro lado, a literatura sobre otimização do planejamento de manutenção preventiva é extensa. Existem diferentes abordagens para a manutenção, com diferentes métodos de otimização e considerações específicas para o contexto da aplicação. Froger et al. (2016), Jonge & Scarf (2020) e Geurtsen et al. (2022) apresentaram uma revisão de literatura sobre problemas de otimização de manutenção.

Froger et al. (2016) descreveram uma revisão da literatura que aborda problemas de programação de manutenção de geradores na indústria elétrica. Segundo os autores, uma das características frequentes desse problema é a inclusão de restrições de janela de tempo para a realização da manutenção. Jonge & Scarf (2020) revisaram a otimização das manutenções. O artigo descreve termos comumente usados na modelagem, na classificação e em soluções. Ele se encerra com os principais desenvolvimentos durante o período de revisão e com as potenciais direções de pesquisa futuras. Um aspecto relevante da conclusão da revisão é que as situações práticas inspiram a literatura. No entanto, apenas alguns artigos aplicaram seus modelos a um estudo de caso com dados reais. Vale ressaltar que, em nosso trabalho,

os dados são obtidos a partir de situações reais do dia a dia da indústria de mineração.

Geurtsen et al. (2022) revisaram problemas que integram a programação da produção com atividades de manutenção. Sua pesquisa inclui uma seção dedicada a problemas de programação de máquinas paralelas com janelas de tempo de manutenção. Um aspecto mencionado neste trabalho é o pequeno número de artigos que abordam a resolução de problemas de programação de manutenção, de sequenciamento de produção e de programação de recursos de maneira integrada.

A literatura sobre aplicações do planejamento de manutenção abrange uma variedade de áreas, como a indústria elétrica, a manutenção ferroviária, a indústria de bens de consumo, a manutenção de aeronaves, as linhas de transmissão e assim por diante. Rodríguez et al. (2018) estudam o problema de programação de manutenção de geradores em sistemas hidrelétricos, que busca maximizar a produção de eletricidade e minimizar os custos de manutenção. Chansombat et al. (2019) apresentam um modelo para resolver simultaneamente o problema de programação integrada de produção e de manutenção preventiva na indústria de bens de capital. Nesse problema integrado, cada manutenção preventiva possui uma janela de tempo e pode ser programada a qualquer instante nela. Portanto, o horário de início da manutenção faz parte da solução para o problema integrado. Shaukat et al. (2020) descreveram um problema em que a manutenção preventiva deve ser programada dentro de janelas de tempo definidas pelo cronograma de desligamento de aeronaves. Considerando que cada manutenção possui um horário de liberação e de vencimento para o qual deve ser programada, o objetivo do problema é minimizar o desvio das janelas de tempo, sem exceder a capacidade de recursos disponíveis (número de equipes de trabalho). Eles propuseram um modelo de programação matemática inteira mista, capaz de encontrar a solução ótima para todas as instâncias reais em menos de 10 minutos, em média.

Schrotenboer et al. (2020) introduziram o problema de transporte de frota de manutenção estocástica para parques eólicos *offshore* com o objetivo de minimizar custos. O problema determina o planejamento de médio prazo para a manutenção de múltiplos parques eólicos, considerando a incerteza nos trabalhos de manutenção e as condições climáticas. Eles estudaram três cenários: (i) realizar todos os trabalhos de manutenção; (ii) permitir uma fração dos trabalhos não programados; e (iii) incentivar a realização da manutenção com rapidez. Embora variações do problema sejam estudadas, uma característica que permanece em todos os cenários é que cada manutenção só pode ser programada dentro de uma janela de tempo. Li et al. (2022) investigaram a otimização bi-objetivo para problemas integrados de alocação de berços e atribuição de guindastes de cais com atividades de manutenção preventiva. Os dois objetivos são: (i) minimizar o tempo total de parada das embarcações e (ii) minimizar o custo das penalidades de manutenção de guindastes de cais antecipadas e tardias. O problema utiliza o conceito de janela de tempo flexível, que permite que a manutenção seja iniciada ou concluída fora da janela. No entanto, há uma penalidade na função objetivo por cada adiantamento ou atraso. Para resolver o problema, eles formularam um modelo de programação linear inteira e uma heurística iterativa bifásica

baseada em restrições ϵ.

Rodríguez et al. (2021) estenderam o modelo de Rodríguez et al. (2018) para uma abordagem estocástica e, também, propuseram um algoritmo exato para a solução do problema. Woller et al. (2023) assumiram uma restrição semelhante a um conjunto de manutenções de uma máquina que não podem ser programadas simultaneamente, associada a um problema de programação de manutenção em linhas de transmissão. Nesse problema, a manutenção não pode ser programada simultaneamente com outras atividades, devido à proximidade física das linhas de transmissão. Zhang et al. (2023) estudaram o problema de otimização da programação de manutenção para minimizar a perda geral de energia durante o tempo de inatividade de um parque eólico *offshore* causada pela manutenção preventiva. O artigo apresenta uma formulação de otimização não linear inteira mista que garante que a manutenção de cada turbina seja restrita a um intervalo de tempo previamente especificado.

Os principais artigos que motivaram o desenvolvimento desta tese foram Aquino et al. (2018a,b, 2019). Esses trabalhos abordam o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (*Long-Term Preventive Maintenance Order Scheduling Problem* - LTPMOSP), proposto com base em um estudo de caso de planejamento de manutenção preventiva de uma unidade de beneficiamento de minério de ferro no Brasil. O problema visa à otimização do planejamento de um conjunto de manutenções preventivas em um horizonte de planejamento de 52 semanas. O objetivo é minimizar o número de equipes de manutenção utilizadas e maximizar o número de manutenções preventivas efetivamente executadas e, desta forma, minimizar a necessidade de manutenções corretivas. Para solucionar este problema, Aquino et al. (2018a) introduziu uma formulação de programação linear inteira mista. No entanto, para gerar soluções factíveis para as instâncias de larga escala, foram implementados algoritmos meta-heurísticos baseados nos métodos *Simulated Annealing* (Aquino et al., 2018a), *Variable Neighborhood Search* (Aquino et al., 2018b) e *Biased Random-Key Genetic Algorithm* (Aquino et al., 2019). As soluções geradas pelos algoritmos desenvolvidos foram capazes de produzir um planejamento melhor do que o empregado pela empresa. Santos et al. (2022) desenvolveu novas abordagens de solução baseadas nos métodos *Iterated Local Search (ILS)*, *Simulated Annealing* e GRASP, com o intuito de melhorar os limites superiores obtidos por Aquino et al. (2019). As abordagens atingem o objetivo, com destaque para o algoritmo ILS, que apresenta o melhor desempenho. Os autores destacam uma melhora de $40,5\%$ no resultado obtido na instância real em relação ao trabalho de Aquino et al. (2019). Recentemente, Coelho et al. (2026) estudaram uma abordagem estocástica do LTPMOSP. Eles propuseram um *simheuristic-based algorithm* capaz de considerar a incerteza no tempo de processamento das ordens de manutenção. Nos experimentos foram testados três cenários com diferentes níveis de incerteza quanto ao tempo das ordens de manutenção. Os resultados mostram que o algoritmo pode ser usado para apoiar a tomada de decisões em diferentes ambientes industriais, desde aqueles com baixa variabilidade na duração do trabalho até aqueles com alta variabilidade.

(Magnoga & Cota, 2023)

Tabela 2.5: Resumo das características comuns dos problemas abordados nos artigos citados

Artigos	Características										
	(i)	*(ii)*	*(iii)*	*(iv)*	*(v)*	*(vi)*	*(vii)*	*(viii)*	*(ix)*	*(x)*	*(xi)*
Chansombat et al. (2019)	✓				✓						✓
Li et al. (2022)	✓				✓				✓	✓	
Oğuz et al. (2010); Silva et al. (2018) de Weerdt et al. (2021); Tarhan & Oğuz (2021, 2022)			✓		✓			✓			
Rodríguez et al. (2018, 2021)	✓				✓				✓		
Rojanasoonthon & Bard (2005, 2006)				✓	✓			✓			
Shaukat et al. (2020)		✓			✓					✓	
Schrotenboer et al. (2020)	✓				✓					✓	
Wang et al. (2015); Emami et al. (2016) Emami et al. (2017); Ou & Zhong (2017) Wu et al. (2018); Liu & Lu (2020) Naderi & Roshanaei (2020)				✓				✓			
Woller et al. (2023)		✓				✓	✓				✓
Wu et al. (2018)				✓			✓	✓			
Zhang et al. (2019)	✓				✓				✓		
Zhang et al. (2023)	✓				✓						✓
Aquino et al. (2018a,b, 2019); Santos et al. (2022); Coelho et al. (2026)		✓		✓	✓	✓	✓	✓			✓
Esta tese		✓		✓	✓	✓	✓	✓			✓

A Tabela 2.5 resume as características comuns dos problemas descritos nos artigos citados nesta seção e na Seção 2.1.1 relacionados ao LTPMOSP. A primeira coluna identifica os artigos. As colunas seguintes referem-se a (*i*) problemas com uma solução integrada entre programação de manutenção e a aplicação; (*ii*) problemas de programação de manutenção; (*iii*) ambientes com uma única máquina; (*iv*) ambientes com máquinas paralelas; (*v*) restrições de janela de tempo; (vi) manutenções que não podem ser programadas para serem processadas simultaneamente; (*vii*) manutenções que exigem um conjunto específico de máquinas e equipes para serem executadas; *(viii)* problemas nos quais todas as manutenções não precisam necessariamente serem executadas, ou seja, algumas podem ser rejeitadas; *(ix)* problemas de curto prazo; *(x)* problemas de médio prazo; *(xi)* problemas de longo prazo. O símbolo ✓indica se a característica da coluna é considerada no problema do artigo referenciado. Os artigos na Tabela 2.5 são classificados em ordem alfabética. No caso de artigos agrupados, a primeira publicação foi considerada para fins de classificação.

A Tabela 2.5 mostra que a maioria dos artigos citados nessa seção tem uma característica em comum com o LTPMOSP, que é a restrição de janela de tempo para execução das manutenções preventivas. Essa restrição aparece frequentemente na literatura por estar fortemente relacionada à manutenção preventiva. É possível encontrar outras características associadas à manutenção preventiva nesses artigos; no entanto, são específicas da aplicação estudada. Woller et al. (2023) é o único artigo citado que considera um problema com a característica (*vi*), além de Aquino et al. (2018a,b, 2019); Santos et al. (2022) e esta tese. Todos esses estudos citados na Seção 2.2 têm a característica comum do LTPMOSP: a possibilidade de alguns trabalhos de manutenção serem rejeitados. Essa característica difere de grande parte da literatura sobre problemas de escalonamento em ambiente de máquina, que trata de problemas em que todos os trabalhos devem ser processados. A restrição de janela de tempo também aparece em Rojanasoonthon & Bard (2005, 2006) e em artigos citados sobre problemas de máquina única. Portanto, é possível encontrar artigos na literatura que estudam problemas com uma ou mais características semelhantes ao LTPMOSP, mas não todas. Até onde sabemos, exceto Aquino et al. (2018a,b, 2019) e

Santos et al. (2022), nenhum outro estudo considera o problema abordado nesta tese.

2.3 Problemas de sequenciamento com multi-habilidades

Afshar-Nadjafi (2021) apresenta uma revisão dos anos 2000 a 2020 sobre pesquisas que estudaram problemas de sequenciamento com multi-habilidades. Dentre as pesquisas revisadas por ele, verifica-se que a multi-habilidade foi abordada de diferentes formas em problemas de sequenciamento. Um dos problemas mais citados pelo autor, devido às diferentes abordagens de multi-habilidade, é o Problema de Programação de Projeto com Restrições de Recurso. Ele consiste, de forma simples, em programar um conjunto de atividades sem que o nível dos recursos seja violado. Uma das variações desse problema considera a hipótese de multi-habilidade. Nela, cada recurso possui várias habilidades necessárias para realizar uma atividade. Almeida et al. (2018) e Almeida et al. (2019) estudaram a versão desse problema que minimiza o *makespan*. Zheng et al. (2017) e Lin et al. (2020) investigaram um problema semelhante, em que os recursos dominam habilidades de diferentes níveis, e uma atividade requer recursos que dominem habilidades em pelo menos um nível específico.

Karam et al. (2017) e Anoshkina & Meisel (2020) abordaram problemas em que os trabalhadores possuem multi-habilidade com diferentes níveis de desempenho. Chen & Gu (2017) e Chen et al. (2020) estudaram um problema multiobjetivo de programação de projetos e de atribuição de equipes com multi-habilidade. Cada projeto contém tarefas que exigem habilidades diversas. Além disso, o nível de habilidade das equipes pode ser aprimorado com a prática. Isso significa que as equipes ficam mais eficientes e requerem menos tempo para executar as tarefas dos projetos. Barghi & Sikari (2022) investigam uma versão do problema mencionado, na qual se considera o tempo de *setup*. Além disso, nessa versão, os projetos requerem multi-habilidade e um ou mais funcionários com a mesma habilidade para serem executados.

Em uma linha de pesquisa semelhante, Qin et al. (2016) investigam o problema *Multi-skilled Workforce Scheduling*, em que os funcionários têm múltiplas habilidades e a produtividade varia com o aprendizado de conhecimento, conforme as tarefas que lhes são atribuídas. Henao et al. (2015) investigaram um problema que envolve a projeção de um plano de treinamento de funcionários, no qual se decidem quais funcionários terão multi-habilidades e quais habilidades cada um deve possuir.

Walter & Zimmermann (2012), por sua vez, investiga um problema de programação de equipes que busca minimizar o tamanho médio das equipes. Para um determinado conjunto de trabalhadores e de tarefas que requerem várias habilidades, é necessário encontrar uma equipe de trabalhadores para cada tarefa que tenha todas as habilidades necessárias para executá-la. Liang et al. (2019) investigam um problema *Multi-Skilled Task Assignment* aplicado a redes rodoviárias, no qual as tarefas requerem multi-habilidades para serem

processadas e os funcionários que as executam também possuem multi-habilidades. Além disso, uma tarefa pode ser executada por mais de um funcionário, caso ele não tenha todas as habilidades que ela exige. Song et al. (2020) também estudam um problema de atribuição de tarefas de multi-habilidades, em que as tarefas requerem a colaboração de um conjunto de trabalhadores com diferentes habilidades. Wang et al. (2021); You & Hsieh (2021) e Liu et al. (2022) seguem a mesma linha dos demais artigos citados neste parágrafo, em que as tarefas requerem multi-habilidade e mais de um funcionário para serem executadas.

2.4 Problemas de planejamento de manutenções preventivas com multi-habilidades

A literatura sobre problemas de planejamento de manutenção preventiva que consideram multi-habilidades é restrita. A seguir, são revisados os artigos que mais se aproximam da linha de pesquisa do LTPMOSP-MS.

Pessan et al. (2007) estudam um caso real de programação de manutenções preventivas que envolve multi-habilidade. Para serem processadas, as manutenções exigem uma ou mais habilidades específicas, que não podem ser desempenhadas por todos os profissionais disponíveis. Além disso, cada profissional pode ter mais de uma habilidade, porém só pode exercê-la uma por vez. Para solucionar o problema, os autores desenvolvem um modelo matemático. Çakirgil et al. (2020) propuseram uma solução integrada para um problema multiobjetivo de roteirização e de programação de mão de obra. Para que as tarefas, que são manutenções preventivas ou corretivas, sejam executadas pelas equipes, exigem-se múltiplas habilidades e diferentes níveis de proficiência. Além disso, as tarefas possuem janelas de tempo e prioridade para serem executadas. Eles desenvolvem um modelo de programação mista para encontrar a fronteira de Pareto. No entanto, em instâncias reais, os autores implementaram um algoritmo de dois estágios. Bocewicz et al. (2023) investigaram um problema de programação envolvendo a alocação de funcionários com multi-habilidades em manutenções preventivas. No entanto, o estudo considera os efeitos de esquecimento e de aprendizagem de habilidade ao longo do horizonte de planejamento. O efeito do esquecimento considera o aumento do tempo para um profissional executar a manutenção de acordo com o tempo que ele fica sem usar a habilidade associada à manutenção. Enquanto, aprendizagem considera a redução do tempo para um profissional executar a manutenção de acordo com tempo que ele mais usa a habilidade associada a manutenção.

Embora os trabalhos revisados nas Seções 2.2, 2.3 e 2.4, que envolvem problemas de planejamento de manutenções preventivas e multi-habilidade, tenham estudado problemas semelhantes ao LTPMOSP-MS, nenhum deles, segundo o conhecimento do autor desta tese, abordou o problema de planejamento de manutenções preventivas e multi-habilidades como aqui realizado. Aquino et al. (2018a,b, 2019) e a Tabela 2.5 apontam que não há, na literatura, um problema que reúna, de forma conjunta, todas as características do LTPMOSP.

Como o LTPMOSP-MS é uma extensão do LTPMOSP, ele é introduzido na literatura pela presente tese, o que constitui uma de suas principais contribuições.

2.5 Métodos heurísticos

Os métodos exatos podem ser inviáveis para soluções de problemas de natureza $\mathcal{NP} - dif cil$ em termos de tempo de processamento e esforço computacional. Para contornar isso, são aplicados métodos heurísticos, capazes de encontrar soluções "boas" em um tempo e esforço computacionais viáveis. A heurística é uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável. No entanto, essa solução não tem garantia de otimalidade, bem como de garantir quão próxima ela está da solução ótima.

Em geral, heurísticas são muito específicas para o problema que visam resolver. Assim, foram desenvolvidos procedimentos heurísticos que apresentam uma estrutura teórica, permitindo sua aplicação em diferentes problemas. Esses procedimentos são conhecidos na literatura como meta-heurísticas. Dentre a variedade de meta-heurísticas existentes na literatura, esta tese aborda a GRASP. A seguir, apresenta-se uma breve descrição desse algoritmo.

2.5.1 *Greedy randomized adaptative search procedure* (GRASP)

O GRASP (Feo & Resende, 1989, 1995; Resende & Ribeiro, 2010, 2016) é uma meta-heurística multi-início (*multi-start*) utilizada na solução de problemas de otimização combinatória. Este algoritmo consiste em um processo iterativo, sendo que cada iteração é composta por duas fases: (i) uma fase construtiva e (ii) uma fase de busca local. A fase construtiva gera uma solução inicial para o problema. Na fase de busca local a solução é refinada, através da exploração da vizinhança da solução.

O Algoritmo 1 descreve, de forma genérica, os passos básicos da meta-heurística GRASP. O algoritmo inicia o contador *it* do número de iterações com o valor zero. Em seguida, o procedimento entra no seu laço principal, que consiste em, a cada iteração, criar uma nova solução inicial e, em seguida, realizar um procedimento de busca local, na tentativa de obter soluções de melhor qualidade para o problema. Por meio do procedimento *atualizaSolucao*, a solução obtida após o refinamento é comparada à melhor gerada até o momento. Caso a nova solução obtida apresente um valor melhor da função objetivo, a melhor solução é atualizada. Atingido o número de iterações previamente definido, o algoritmo retorna a melhor solução encontrada.

Na criação da solução inicial, os elementos candidatos são todos os que podem ser incorporados à solução parcial em construção, sem violar nenhuma restrição do problema. A definição do próximo elemento a ser incorporado na solução é determinada pela avaliação de todos os elementos candidatos, de acordo com uma função de avaliação. A avaliação

Algoritmo 1: Meta-heurística GRASP

Entrada: $GRASPmax, \alpha$

```
it ⟵ 0
while (it ≤ GRASPmax) do
    solucao ⟵ Construcao(α)
    solucao ⟵ BuscalLocal(solucao)
    atulizaSolucao(solucao, melhorSolucao)
    it ⟶ it + 1
end
Retorne s*
```

do elemento por essa função resulta na criação de uma lista restrita de candidatos (LRC), ou, em inglês, *Restricted Candidate List*. Esta lista é composta pelos melhores elementos, segundo a avaliação. O elemento incorporado na solução parcial é aleatoriamente selecionado dentre os contidos na lista restrita, o que constitui um aspecto probabilístico da heurística. Uma vez que o elemento é selecionado e adicionado à solução parcial, a lista de candidatos é atualizada. Essa estratégia é similar à heurística semi-gulosa proposta por Hart & Shogan (1987), que também possui uma abordagem multi-reinício baseada em construções gulosas aleatórias, porém, sem a utilização de busca local.

Hart & Shogan (1987) e Feo & Resende (1989) propuseram duas formas de construir uma lista restrita. Em Hart & Shogan (1987), a criação baseia-se na cardinalidade, ou seja, as primeiras k melhores soluções são consideradas. Por outro lado, em Feo & Resende (1989), ela é baseada no valor da função objetivo das soluções gulosas obtidas. Em Resende et al. (2000), os elementos inseridos na lista são todos cujo valor da função objetivo esteja entre $\alpha(c_{\max} - c_{\min})$ do valor guloso, sendo $c_{\max} = \max\{c(e) \,|\, e \in C\}$ e $c_{\min} = \min\{c(e) \,|\, e \in C\}$, em que C é o conjunto de elementos candidatos. Para um problema de minimização, a lista restrita é composta por todas as soluções em que

$$LRC = \{e \in C \,|\, c(e) \leq c_{\min} + \alpha(c_{\max} - c_{\min})\}\,.$$

Vale ressaltar que, se $\alpha = 0$, a heurística se torna puramente gulosa; por outro lado, se $\alpha = 1$, a busca é puramente aleatória. As implementações iniciais do GRASP utilizaram um valor fixo de α; no entanto, em pesquisas posteriores, foram desenvolvidas abordagens de autoajuste do parâmetro α, como as propostas por Prais & Ribeiro (2000a,b).

A fase de busca local consiste em iterar sobre a vizinhança da solução corrente, em busca de uma solução melhor. As vizinhanças são geradas por modificações realizadas na solução corrente, o que se denomina movimento. No método de descida com a estratégia *best improvement*, a busca analisa todos os vizinhos e escolhe a solução de melhor qualidade entre eles. No entanto, este método pode ser exaustivo e computacionalmente caro, dependendo da dimensão da vizinhança. Para evitar isso, pode-se aplicar a estratégia *first improvement*. Nesta estratégia, a busca é interrompida quando um vizinho melhor é encontrado. Um

outro método é o *random descent/uphill*, que consiste em analisar um vizinho qualquer e o aceitar somente se ele for melhor que a solução corrente; caso não seja, a solução corrente permanece inalterada e outro vizinho é gerado. A busca iterativa é interrompida após um número fixo de iterações sem melhora na solução. Para o método *random descent/uphill*, não há garantia de que a solução final seja um ótimo local.

2.6 Considerações parciais

Neste capítulo foram revisados os principais conceitos, definições, trabalhos e formulações matemáticas sobre o problema de sequenciamento de produção e a meta-heurística GRASP. Além disso, foi revisada uma série de pesquisas envolvendo problemas de planejamento de manutenções preventivas e multi-habilidade. Por fim, destacou-se que os problemas abordados nesta tese são novos na literatura. O LTPMOSP foi introduzido em Aquino et al. (2018a) e investigado em Aquino et al. (2018b, 2019), Santos et al. (2022) e Coelho et al. (2026). No entanto, o LTPMOSP-MS é introduzido nesta tese.

Nos capítulos 3 e 4, serão formalmente definidos o LTPMOSP e o LTPMOSP-MS, respectivamente.

Capítulo 3

Problema de planejamento de ordens de manutenção de longo prazo

Este capítulo é dedicado ao estudo do Problema de Planejamento de Ordens de Manutenções Preventivas de Longo Prazo (LTPMOSP). Inicialmente, a Seção 3.1 descreve formalmente o LTPMOSP. A Seção 3.2 apresenta um exemplo descritivo do LTPMOSP, usando, para tal, uma instância de pequena dimensão, de modo a facilitar a compreensão do problema. A Seção 3.3 apresenta a formulação matemática para o LTPMOSP proposta por Aquino et al. (2018a). As Seções 3.4 e 3.5 introduzem as formulações do LTPMOSP desenvolvidas para esta tese. A primeira formulação é baseada em variáveis indexadas no tempo e a segunda combina variáveis indexadas no tempo com variáveis de precedência. Em seguida, as dimensões das duas formulações matemáticas com a formulação de Aquino et al. (2018a) são comparadas na Seção 3.6.

Devido ao fato de o problema pertencer à classe $\mathcal{NP}$-difícil, os métodos exatos podem não ser adequados para solucionar instâncias de grande porte, por requererem altos valores de tempo computacional e de memória computacional para sua resolução. Para contornar essa desvantagem, é apresentado um algoritmo baseado na meta-heurística GRASP (Feo & Resende, 1995) na Seção 3.7.

Para comparar e analisar o desempenho dos métodos de solução propostos nesta tese é realizado um extenso experimento computacional. Os resultados são apresentados, analisados e discutidos na Seção 3.8. Por fim, a Seção 3.9 apresenta conclusões parciais com relação aos resultados aqui expostos.

3.1 Descrição do problema

O Problema de Manutenções Preventivas de Longo Prazo (*Long-Term Preventive Maintenance Order Scheduling Problem* -LTPMOSP) pode ser formalmente definido da seguinte forma. Seja um conjunto I de ordens de manutenções que devem ser executadas, um conjunto K de equipes de trabalho disponíveis para executar essas ordens de manutenções e

um conjunto M de máquinas nas quais essas manutenções serão realizadas.

Cada manutenção $i \in I$ pode ser executada apenas por um subconjunto $\Gamma_i \subseteq K$ de equipes que possuem a habilidade hab_i requerida para realizar a manutenção. Além disso, cada manutenção $i \in I$ requer um tempo de execução p_i e deve ser realizada na máquina $m = e_i \in M$ no período de tempo representado pela janela de tempo $[\alpha_i, \beta_i]$. As manutenções podem ser executadas em qualquer instante da janela de tempo, desde que iniciem e finalizem nos limites da mesma. Caso a manutenção $i \in I$ não seja finalizada dentro da janela, ela é classificada como não executada e penalizada com o valor w_i.

Cada equipe de trabalho $k \in K$ tem apenas uma habilidade hab_k (por exemplo, mecânica, elétrica, etc.) para a execução das manutenções e ela está disponível somente no intervalo de tempo $[0, h_k]$. Além disso, seja o subconjunto $\Omega_k \subseteq I$, que denota o conjunto de manutenções que a equipe de trabalho $k \in K$ tem habilidade para realizar. Por fim, seja o subconjunto $\Phi_m \subseteq I$, que denota o conjunto de manutenções a serem realizadas na máquina $m \in M$.

O LTPMOSP tem, como objetivo, minimizar o número de equipes de trabalho ativas e o somatório das penalidades das manutenções não executadas, satisfazendo às seguintes restrições:

(i) se executada, cada manutenção deve ser designada a exatamente uma equipe de trabalho;

(ii) cada equipe de trabalho pode executar somente uma manutenção por vez;

(iii) cada máquina $m \in M$ não pode ter mais de uma manutenção sendo executada simultaneamente;

(iv) cada equipe $k \in K$ está disponível somente no intervalo $[0, h_k]$;

(v) cada manutenção $i \in I$ deve ser executada dentro de sua janela de tempo $[\alpha_i, \beta_i]$ previamente estabelecida; e

(vi) cada manutenção $i \in \Omega_k \subseteq I$ requer uma equipe $k \in \Gamma_i \subseteq K$ que tenha a habilidade específica para realizá-la.

3.2 Exemplo para descrição do LTPMOSP

Esta seção apresenta uma ilustração do LTPMOSP usando uma instância teste de pequena dimensão. Os dados de entrada referentes a este exemplo são apresentados nas Tabelas 3.1 e 3.2. A Tabela 3.1 mostra, para cada manutenção i, a habilidade hab_i necessária para executar a manutenção e demais parâmetros, de acordo com a notação apresentada na definição do LTPMOSP. A Tabela 3.2 apresenta, para cada equipe k, a habilidade hab_k que a equipe de trabalho possui, e o instante final h_k, em horas, de disponibilidade da equipe.

Tabela 3.1: Dados de entrada das tarefas.

i	hab_i	e_i	α_i	β_i	p_i	w_i
1	2	213	3	9	4	72
2	2	199	2	7	3	132
3	1	199	1	5	2	20
4	1	342	2	6	3	30
5	2	213	0	10	5	212
6	1	213	0	3	2	168
7	2	199	4	8	3	40

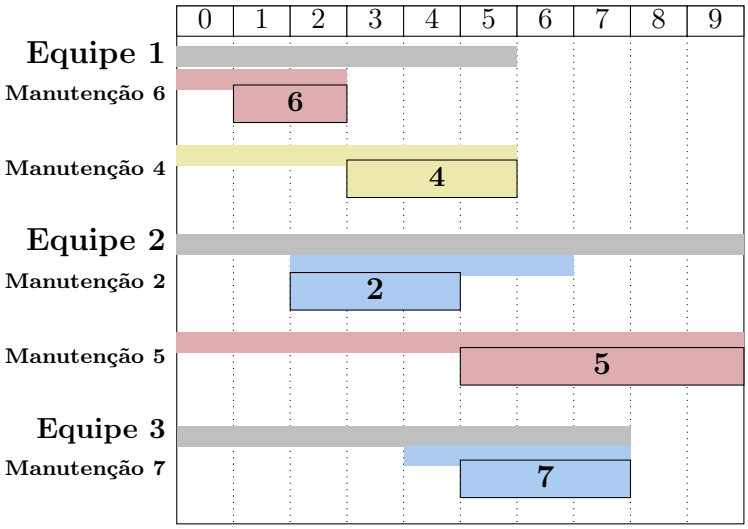
Tabela 3.2: Dados de entrada das equipes de trabalho.

k	hab_k	h_k
1	1	6
2	2	10
3	2	8

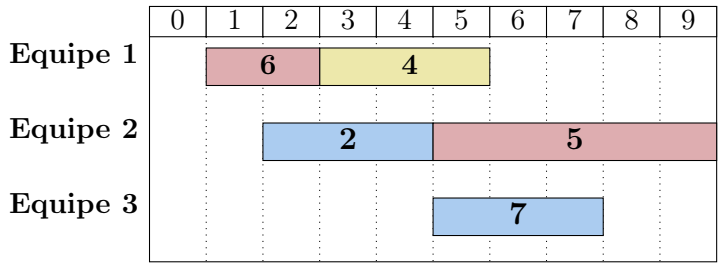
A Figura 3.1 ilustra, através de um gráfico de Gantt, o sequenciamento das manutenções. Esta figura está dividida em três partes. Na sua parte superior está a linha do tempo, na qual cada número representa uma hora. Na Figura 3.1(a), as barras em cor cinza representam a janela de tempo em que cada equipe pode trabalhar, enquanto que a barra na parte superior da cada manutenção, e com a mesma cor, corresponde à sua janela de tempo.

A Figura 3.1(b) ilustra a programação das manutenções para as equipes de trabalho. Observa-se que a equipe de trabalho 1 executa as manutenções 6 e 4 em sequência, a equipe 2 executa as manutenções 2 e 5 nesta sequência e a equipe 3 executa apenas a manutenção 7. Nota-se, também, que as manutenções 6 e 4 estão alocadas à equipe de trabalho 1, pois ela possui a habilidade necessária para executar estas duas manutenções. As demais manutenções estão alocadas às equipes de trabalho 2 e 3, pois requerem a habilidade 2.

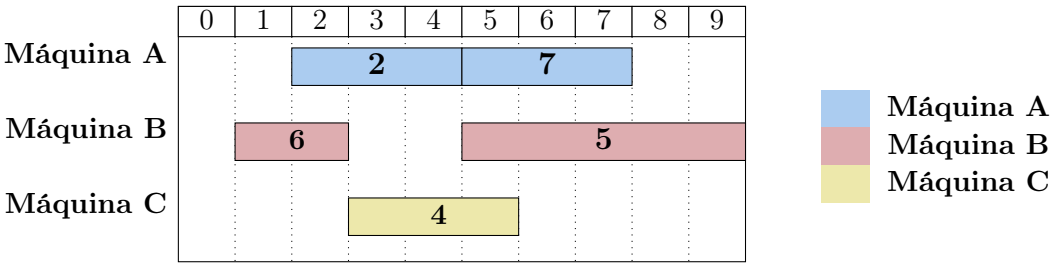
A Figura 3.1(c) mostra a alocação das manutenções às máquinas. Na parte inferior da figura, as três cores diferentes identificam em qual máquina será realizada cada manutenção. Nesta parte, as manutenções 2 e 7, nesta sequência, estão alocadas à máquina 199; as manutenções 6 e 5 estão alocadas à máquina 213 e a manutenção 4 está alocada à máquina 342. Evidentemente, os tempos de início e conclusão das manutenções são os mesmos das equipes de trabalho e máquinas. Por esta figura, nota-se que existe uma sequência de manutenções nas equipes e nas máquinas, que é uma das principais características que levam ao aumento da complexidade do LTPMOSP. Note, por fim, que, na solução apresentada na Figura 3.1, não foi possível alocar as manutenções 1 e 3 em nenhuma equipe de trabalho; logo, elas não são executados.



(a) Sequenciamento das manutenções com janelas de tempo.



(b) Programação das manutenções nas equipes de trabalho.



(c) Programação das manutenções nas máquinas.

Figura 3.1: Ilustração de uma solução factível para o LTPMOSP.

.

3.3 Formulação matemática da literatura

A formulação matemática proposta por Aquino et al. (2018a), aqui denotada como formulação NET (*Formulation based on Network variables*), que é baseada em variáveis de rede, utiliza os conjuntos e os parâmetros apresentados na Seção 3.1. Além disso, esta formulação utiliza os parâmetros B' e B'' para representar números suficientemente grandes. Define-se, então as seguintes variáveis de decisão:

(i) y_k é uma variável binária que é igual a 1 se a equipe de trabalho k é ativada; e 0, caso contrário;

(ii) x_{ik} é uma variável binária que é igual a 1 se a manutenção i for programada para ser executada pela equipe de trabalho k; e 0, caso contrário;

(iii) v_{ij}^{k} é uma variável binária que é igual a 1 se a manutenção i precede imediatamente a manutenção j programada para ser executada pela equipe de trabalho k; e 0, caso contrário. Neste caso, considera-se uma manutenção fictícia denotada por 0;

(iv) r_{ij} é uma variável binária que é igual a 1 se a manutenção i precede a manutenção j no uso da mesma máquina quando ambas as manutenções são executadas por equipes diferentes; e 0, caso contrário;

(v) c_{ik} é uma variável não negativa que representa o tempo de conclusão da manutenção i pela equipe de trabalho k.

A formulação matemática (NET) proposta por Aquino et al. (2018a) é dada, assim, por:

$$(NET) \quad \min \quad z = \sum_{k \in K} y_k + \sum_{i \in I} w_i \left(1 - \sum_{k \in \Gamma_i} x_{ik}\right) \tag{3.1}$$

$$\textbf{suj. a:} \sum_{k \in \Gamma_i} x_{ik} \leq 1 \qquad \forall i \in I \tag{3.2}$$

$$\sum_{i \in \Omega_k \cup \{0\} \setminus \{j\}} v_{ij}^{k} = x_{jk} \qquad \forall j \in I, k \in \Gamma_j \tag{3.3}$$

$$\sum_{j \in \Omega_k} v_{0j}^{k} = y_k \qquad \forall k \in K \tag{3.4}$$

$$\sum_{i \in \Omega_k \cup \{0\} \setminus \{l\}} v_{il}^{k} = \sum_{j \in \Omega_k \cup \{0\} \setminus \{l\}} v_{lj}^{k} \qquad \forall k \in K, l \in \Omega_k \tag{3.5}$$

$$c_{0k} = 0 \qquad \forall k \in K \tag{3.6}$$

$$c_{jk} \geq c_{ik} + p_j - B'_{ij}(1 - v_{ij}^{k}) \qquad \forall k \in K, i \in \Omega_k \cup \{0\}, j \in \Omega_k \tag{3.7}$$

$$c_{ik} \geq (\alpha_i + p_i) x_{ik} \qquad \forall k \in K, i \in \Omega_k \tag{3.8}$$

$$c_{ik} \leq \beta_i \qquad \forall k \in K, i \in \Omega_k \tag{3.9}$$

$$c_{jk'} \leq c_{ik} - p_i + B''_{ij} r_{ij} \qquad \forall k \in K, \forall k' \in K, i \in \Omega_k, j \in \Omega_{k'}, k \neq k', i < j, e_i = e_j \tag{3.10}$$

$$c_{jk'} \geq c_{ik} + p_j - B'_{ij}(1 - r_{ij}) \qquad \forall k \in K, \forall k' \in K, i \in \Omega_k, i \in \Omega_{k'}, k \neq k', i < j, e_i = e_j \tag{3.11}$$

$$c_{ik} \leq h_k \qquad \forall k \in K, i \in \Omega_k \tag{3.12}$$

$$v_{ij}^{k} \in \{0, 1\} \qquad \forall k \in K, i \in \Omega_k \cup \{0\}, j \in \Omega_k \cup \{0\} \tag{3.13}$$

$$x_{ik} \in \{0, 1\} \qquad \forall k \in K, i \in \Omega_k \tag{3.14}$$

$$y_k \in \{0, 1\} \qquad \forall k \in K \tag{3.15}$$

$$r_{ij} \in \{0, 1\} \qquad \forall i \in I, j \in I | i < j, e_i = e_j \tag{3.16}$$

$$c_{ik} \geq \{0, 1\} \qquad \forall k \in K, i \in \Omega_k \tag{3.17}$$

A função objetivo (3.1) minimiza o número de equipes de trabalho ativadas (primeiro termo) e a soma das penalidades das manutenções não executadas (segundo termo). As restrições (3.2) garantem que, caso uma manutenção seja realizada, ela deve ser executada por apenas uma equipe. As restrições (3.3) asseguram que caso uma manutenção seja executada por uma equipe, ela deve ser precedida por uma manutenção, que pode incluir a manutenção fictícia 0. As restrições (3.4) impõem que uma equipe seja ativada somente se tiver uma ou mais manutenções alocadas. As restrições (3.5) são responsáveis por manter a sequência das manutenções executadas em cada equipe. As restrições (3.6) impõem que o tempo de conclusão da manutenção fictícia seja zero em todas as equipes. Esta manutenção é alocada antes da primeira manutenção executada por cada equipe. As restrições (3.7) garantem que o tempo de conclusão da manutenção i seja menor que o tempo de conclusão da manutenção j, caso a manutenção i preceda a manutenção j na mesma equipe. As restrições (3.8) asseguram que o tempo de conclusão de cada manutenção seja maior que o início da janela mais o tempo de processamento. As restrições (3.9) impõem que o tempo de conclusão de cada manutenção seja menor que o final da janela. As restrições (3.10) e (3.11) são restrições disjuntivas, responsáveis por garantir que cada máquina não tenha mais de uma manutenção sendo executada simultaneamente. As restrições (3.12) asseguram que o instante de conclusão das manutenções alocadas a cada equipe seja menor ou igual à quantidade de horas disponíveis. As restrições (3.13)-(3.17) definem o domínio das variáveis de decisão.

3.4 Formulação baseada em variáveis indexadas no tempo

A formulação baseada em variáveis indexadas no tempo proposta nesta tese, é baseada na formulação de Sousa & Wolsey (1992) para solucionar um problema de sequenciamento de uma única máquina. Neste tipo de formulação, o horizonte de planejamento é dividido em períodos. De uma forma geral, o problema consiste em definir em qual período cada manutenção deve iniciar o seu processamento, sequenciando, deste modo, as manutenções. A principal vantagem das formulações com variáveis indexadas no tempo é que sua relaxação linear fornece um limite inferior forte (no caso dos problemas de minimização), enquanto sua maior desvantagem é o grande número de variáveis e os tempos de processamento longos, especialmente quando o horizonte de planejamento tem um grande valor.

Nesta formulação, o conjunto de períodos é denotado por T. A formulação matemática baseada em variáveis indexadas no tempo para o LTPMOSP, denotada como formulação TI, utiliza os conjuntos e parâmetros apresentados na Seção 3.1, e as variáveis de decisão y_k e x_{ik} apresentadas na Seção 3.3. Para garantir que cada manutenção seja executada dentro da sua janela de tempo e no tempo de disponibilidade das equipes, foram adicionadas limitações no índice $t \in T$ das variáveis e das restrições. Para isto, a formulação TI utiliza $\bar{T}_i^k \subseteq T$ para

denotar o subconjunto de períodos em que a manutenção i pode começar a ser executada na equipe k, de forma a respeitar as janelas de tempo da manutenção e a disponibilidade da equipe, ou seja:

$$\bar{T}_i^k = \{t \in T : t \leq h_k - p_i, \alpha_i \leq t \leq \beta_i - p_i\}.$$

Além disso, será utilizado o parâmetro:

$$\bar{f}_{ik}^t = \min\{\bar{t} \in \bar{T}_i^k : \bar{t} \geq t - p_i + 1\}$$

para denotar o menor período de início de execução da manutenção i na equipe k que pode estar em execução no período t. A formulação também utiliza a variável binária u_{ik}^t, de modo que $u_{ik}^t = 1$ se a manutenção i for programada para começar a ser executada no período t pela equipe k; e $u_{ik}^t = 0$, caso contrário, para $i \in I, k \in \Gamma_i, t \in \bar{T}_i^k$.

A formulação TI é baseada na versão clássica proposta por Sousa & Wolsey (1992). No entanto, como a versão clássica considera que todas as manutenções devem ser processadas, foi necessário realizar algumas adaptações na formulação. Além disso, foram feitas modificações para alocação simultânea das manutenções às equipes e às máquinas e para atender as janelas de execução. A formulação TI é dada, portanto, por:

$$(TI) \quad \min \quad z = \sum_{k \in K} y_k + \sum_{i \in I} w_i \left(1 - \sum_{k \in \Gamma_i} x_{ik}\right) \tag{3.18}$$

$$\textbf{suj. a:} \quad (3.2), \quad (3.14) - (3.15) \tag{3.19}$$

$$x_{ik} \leq y_k \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.20}$$

$$\sum_{t \in \bar{T}_i^k} u_{ik}^t \geq x_{ik} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.21}$$

$$\sum_{k \in \Gamma_i} \sum_{t \in \bar{T}_i^k} u_{ik}^t \leq 1 \qquad \forall i \in I \tag{3.22}$$

$$\sum_{i \in \Omega_k} \sum_{f=\bar{f}_{ik}^t}^{t} u_{ik}^f \leq 1 \qquad \forall k \in K, \forall t \in T \tag{3.23}$$

$$\sum_{i \in \Phi_m} \sum_{k \in \Gamma_i} \sum_{f=\bar{f}_{ik}^t}^{t} u_{ik}^f \leq 1 \qquad \forall m \in M, \forall t \in T \tag{3.24}$$

$$u_{ik}^t \in \{0, 1\} \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in \bar{T}_i^k \tag{3.25}$$

A função objetivo (3.18) é igual à função objetivo da formulação NET. As restrições (3.20) impõem que uma equipe de trabalho será ativada somente se tiver uma ou mais manutenções alocadas. As restrições (3.21) associam as variáveis u_{ik}^t e x_{ik}, ao garantir que a manutenção é iniciada apenas na equipe em que ela é alocada. As restrições (3.22) garantem que, caso uma manutenção seja alocada a uma equipe, ela deverá ser iniciada em algum momento, e exatamente uma vez e em um período. As restrições (3.23) garantem que não haverá mais de uma ordem programada para estar em execução em uma mesma

equipe no mesmo período. As restrições (3.24) garantem que não haverá mais de uma ordem programada para estar em execução em uma mesma máquina no mesmo período. As restrições (3.25) definem o domínio das variáveis de decisão.

3.5 Formulação baseada em variáveis indexadas no tempo e variáveis de precedência

A segunda formulação proposta neste trabalho, que é denotada por formulação TIP, é baseada em variáveis indexadas no tempo e em variáveis de precedência. Na formulação TIP, o horizonte de planejamento é dividido em períodos e as manutenções podem ser programadas para serem executadas em um único período ou em vários períodos, dependendo da duração do período e de quando a manutenção é executada. Além disso, esta formulação considera que, em um mesmo período, podem ser alocadas diferentes manutenções para serem executadas na mesma equipe ou na mesma máquina, diferentemente do que ocorre na formulação TI, em que apenas uma manutenção pode ser executada na mesma equipe ou máquina por período. Assim, na formulação TIP, as variáveis indexadas no tempo são responsáveis por indicar se a manutenção está alocada a cada período, enquanto que as variáveis de precedência são responsáveis por sequenciar as manutenções, ou partes delas, dentro dos períodos para garantir a não sobreposição de ordens de manutenção nas equipes e nas máquinas. A vantagem desta formulação é que permite maior flexibilidade de discretização do horizonte de planejamento, podendo resultar em uma formulação de menor dimensão que a formulação TI.

Para a formulação TIP, considere o conjunto de períodos $T = \{0, 1, \ldots, t_{max}\}$. A duração de cada período é definida pelo parâmetro π. Desse modo, o período t se inicia no instante $t\pi$ e se encerra no instante $(t+1)\pi$. 0 parâmetro π é muito importante para a formulação TIP, pois ele afeta o número de períodos em que o horizonte de planejamento é dividido e o número de períodos utilizados para executar cada manutenção. Além disso, esse parâmetro interfere na dimensão da formulação.

A formulação TIP utiliza as seguintes variáveis de decisão:

(i) A variável binária u_{ik}^t é igual a 1 se a manutenção i for alocada à equipe k no período t; caso contrário, ela é igual a 0;

(ii) As variáveis Δ_{ik}^t, que variam entre 0 e π, representam o tempo de execução da manutenção i alocada à equipe k no período t;

(iii) A variável binária ψ_{ij}^t, usada para sequenciar as manutenções dentro de um mesmo período, é igual a 1 se a manutenção i precede imediatamente a manutenção j no período t na mesma equipe; caso contrário, é igual a 0;

(iv) A variável binária μ_{ij}^t é igual a 1 se a manutenção i precede imediatamente a ordem j no período t em uma máquina em que ambas estão associadas ($e_i = e_j$); caso contrário,

é igual a 0;

(v) A variável c_{ik}^t, que varia entre 0 e π, indica o instante de conclusão da manutenção i na equipe k no período t. Se t é o último período no qual a manutenção i foi programada, então o tempo de conclusão dela é $t\pi + c_{ik}^t$;

(vi) A variável binária s_{ik}^t é igual a 1 se a manutenção i é programada nos períodos $t-1$ e t na equipe k; caso contrário, é igual a 0.

(vii) A variável binária f_{ik}^t é igual a 1 se a manutenção i é programada nos períodos t e $t+1$ na equipe k; caso contrário, é igual a 0.

As variáveis s_{ik}^t e f_{ik}^t são utilizadas para garantir que uma manutenção que é executada em dois períodos seguidos tem que finalizar o primeiro dos períodos em execução e iniciar o período seguinte em execução para garantir a não preempção.

Para exemplificar o uso dessas variáveis, considere uma instância com apenas uma equipe e duas manutenções, em que $\pi = 4$, $p_1 = 3$h, $p_2 = 6$h e $T = \{0, 1, 2\}$. Como o tempo de execução da manutenção 1 é menor que a duração do período ($p_i \leq \pi$), uma solução possível é executar toda a manutenção 1 no período 1. Logo, $\Delta_{11}^t = \{0, 3, 0\}$ e $u_{11}^t = \{0, 1, 0\}$ (veja a Figura 3.2a). Outra possibilidade de programação para a manutenção 1 é iniciar sua execução no início da hora 2 no período 0 e finalizar sua execução ao final da hora 4 (primeira hora do período 1). Logo, $\Delta_{11}^t = \{2, 1, 0\}$ e $u_{11}^t = \{1, 1, 0\}$ (Figura 3.2b). Quanto à manutenção 2, como $p_2 > \pi$, é necessário programar a execução desta manutenção em pelo menos dois períodos. Dois exemplos de solução podem ser dados por: (i) $\Delta_{21}^t = \{0, 2, 4\}$ e $u_{21}^t = \{0, 1, 1\}$ (Figura 3.2c), (ii) $\Delta_{21}^t = \{1, 4, 1\}$ e $u_{21}^t = \{1, 1, 1\}$ (Figura 3.2d). Note que $\sum_t \Delta_{ik}^t$ é sempre igual a p_i. A Figura 3.2 ilustra os exemplos de soluções apresentadas para manutenção 1 (em azul) e 2 (em vermelho). A primeira linha desta figura refere-se ao período e a segunda, à hora de execução.

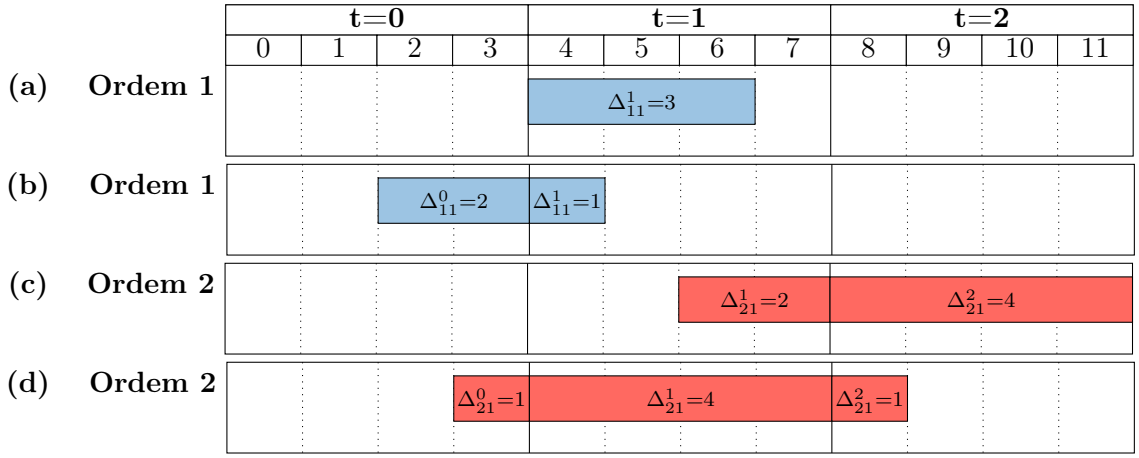


Figura 3.2: Exemplos de soluções para as variáveis Δ_{ik}^t.

Para programar o tempo de processamento Δ_{ik}^t nos períodos e consequentemente as manutenções, são consideradas as variáveis c_{ik}^t e ψ_{ij}^t. A Figura 3.3 mostra um exemplo de

solução com as manutenções 1 e 2 sendo executadas por apenas uma equipe. A solução é $\Delta_{11}^t = \{2,1,0\}$ e $u_{11}^t = \{1,1,0\}$ e $\Delta_{21}^t = \{0,2,4\}$ e $u_{21}^t = \{0,1,1\}$. Para essa solução, tem-se os seguintes valores para c_{ik}^t: $c_{11}^t = \{4,1,0\}$ e $c_{21}^t = \{0,4,4\}$. Além disso, apenas $\psi_{12}^1 = 1$. Como a manutenção 1 inicia no período 0 e finaliza no período seguinte, Δ_{11}^0 é alocado ao final do período, logo $c_{11}^t = 4 = \pi$, e Δ_{11}^1 é alocado no início período, logo $c_{11}^1 = 1 = \Delta_{11}^1$. Portanto, não existe preempção. A mesma explicação é aplicada para os valores de c_{ik}^t para manutenção 2. A variável $\psi_{12}^1 = 1$ determina a sequência das manutenções alocadas no período 1. Através das variáveis c_{ik}^t, os tempos correspondentes às variáveis Δ_{ik}^t são programados nos períodos, de forma que as duas manutenções não são executadas ao mesmo tempo na equipe. As linhas verticais em vermelho na Figura 3.3 destacam os tempos de conclusão de cada Δ_{ik}^t.

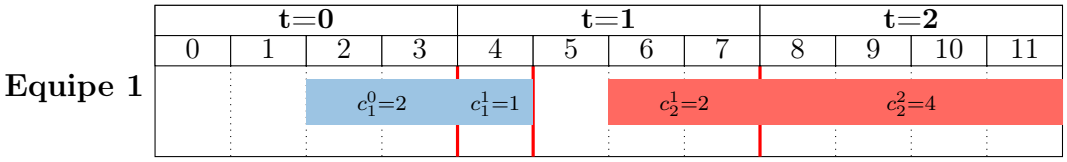


Figura 3.3: Exemplo de solução com duas manutenções alocada em uma equipe.

Seja o mesmo exemplo, agora adicionando-se uma segunda equipe e uma terceira manutenção, em que $p_3 = 1$, que exige uma habilidade que apenas a segunda equipe tenha. Esta terceira manutenção pode ser alocada à segunda equipe (em amarelo na Figura 3.4). Como as três manutenções estão associadas à mesma máquina, elas não podem ser executadas simultaneamente. Para garantir isso, são utilizadas as variáveis μ_{ij}^t. A Figura 3.4 apresenta um exemplo de solução com as três manutenções. Nesta solução, as manutenções 1 e 2 estão alocadas à equipe 1 e possuem os mesmo valores da solução da Figura 3.3, enquanto que para a manutenção 3, temos $\Delta_{32}^t = \{0,1,0\}$, $u_{32}^t = \{0,1,0\}$ e $c_{32}^t = \{0,2,0\}$. Além disso, $\mu_{13}^1 = \mu_{32}^1 = 1$ e as demais variáveis μ_{ij}^t são iguais a 0, indicando que a manutenção 1 precede a manutenção 3 e a manutenção 3 precede a manutenção 2 na máquina. Os valores das variáveis c_i^t e o diagrama de Gantt na Figura 3.4 mostram a programação do tempo de execução das manutenções referente às variáveis Δ_{ik}^t em um período com várias manutenções alocadas à mesma equipe e que estas manutenções não são executadas ao mesmo tempo.

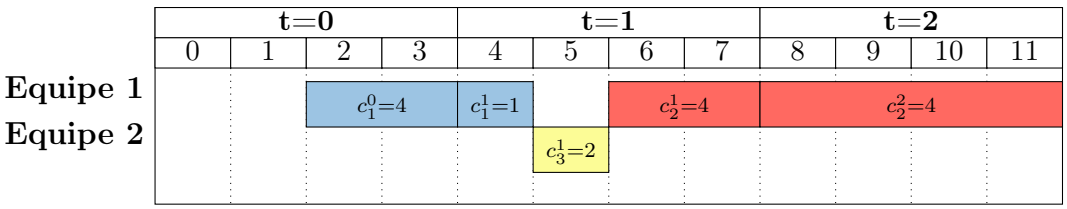


Figura 3.4: Exemplo de solução com três manutenções, associadas à mesma máquina, alocadas em duas equipes.

Para garantir que as manutenções sejam executadas dentro da janela de tempo, são

considerados os parâmetros $t_i^{\alpha}, t_i^{\beta}, \delta_i^{\alpha}$ e δ_i^{β} para cada manutenção i. O parâmetro t_i^{α} está associado ao período de início da janela de tempo da manutenção i, sendo obtido pela expressão $t_i^{\alpha} = \lfloor \frac{\alpha_i}{\pi} \rfloor$. O parâmetro t_i^{β} está associado ao período de término da janela de tempo da manutenção i, sendo dado pela expressão $t_i^{\beta} = \lceil \frac{\beta_i}{\pi} \rceil - 1$. No entanto, o início e o fim da janela de tempo podem não ocorrer exatamente no início ou fim dos períodos. Por exemplo, considere $\pi = 4$ e uma manutenção i com a janela de tempo de $\alpha_i = 3$ e $\beta_i = 10$. Nesse exemplo, a janela inicia após o início do período 0 e se encerra antes do fim do período 2, como mostra a Figura 3.5. Assim, utiliza-se δ_i^{α} para contabilizar o tempo do início do período até o início da janela e utiliza-se δ_i^{β} para contabilizar o tempo do início do período até o fim da janela. O parâmetro δ_i^{α} e δ_i^{β} são dados, respectivamente, pelas expressões $\delta_i^{\alpha} = \alpha_i - t_i^{\alpha}\pi$ e $\delta_i^{\beta} = \beta_i - t_i^{\beta}\pi$. Aplicando os valores da manutenção i nas expressões, no exemplo mencionado, para tornar mais claros os parâmetros, tem-se: $t_i^{\alpha} = 0$, $t_i^{\beta} = 2$, $\delta_i^{\alpha} = 3$ e $\delta_i^{\beta} = 2$. Na Figura 3.5 é possível visualizar os parâmetros δ_i^{α} e δ_i^{β}. De forma semelhante, são utilizados os parâmetros t_k^h e δ_k^h para garantir que as manutenções não sejam executadas depois do tempo limite de trabalho de cada equipe k. Os parâmetros t_k^h e δ_k^h são definidos por $t_k^h = \lceil \frac{h_k}{\pi} \rceil - 1$ e $\delta_k^h = h_k - t_k^h\pi$.

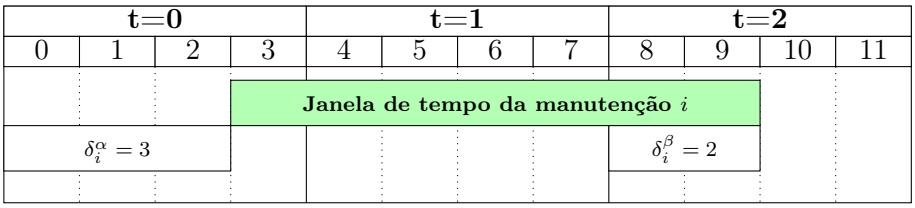


Figura 3.5: Exemplo ilustrando os parâmetros δ_i^{α} e δ_i^{β} para a janela de tempo da manutenção i.

Para a formulação TIP são utilizadas as variáveis de decisão e parâmetros apresentados nesta subseção, os conjuntos e parâmetros apresentados na Seção 3.1 e as variáveis de decisão y_k e x_{ik} apresentadas na Seção 3.3. Assim, a formulação TIP proposta pode ser dada por:

$$(TIP) \quad \min \quad z = \sum_{k \in K} y_k + \sum_{i \in I} w_i \left(1 - \sum_{k \in \Gamma_i} x_{ik}\right) \tag{3.26}$$

$$\textbf{s. a:} \quad (3.2), \quad (3.14) - (3.15), (3.20) \tag{3.27}$$

$$\sum_{k \in \Gamma_i} p_i x_{ik} = \sum_{k \in \Gamma_i} \sum_{t \in T} \Delta_{ik}^t \qquad \forall i \in I \tag{3.28}$$

$$\left(\left\lceil \frac{p_i}{\pi} \right\rceil + 1\right) \sum_{k \in \Gamma_i} x_{ik} \geq \sum_{k \in \Gamma_i} \sum_{t \in T} u_{ik}^t \qquad \forall i \in I \tag{3.29}$$

$$\left\lceil \frac{p_i}{\pi} \right\rceil \sum_{k \in \Gamma_i} x_{ik} \leq \sum_{k \in \Gamma_i} \sum_{t \in T} u_{ik}^t \qquad \forall i \in I \tag{3.30}$$

$$\Delta_{ik}^t \leq \pi u_{ik}^t \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \tag{3.31}$$

$$u_{ik}^t \leq x_{ik} \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \tag{3.32}$$

$$\Delta_{ik}^{t} \leq \pi x_{ik} \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \tag{3.33}$$

$$\Delta_{ik}^{t} \geq (u_{ik}^{t-1} + u_{ik}^{t+1} - 1)\pi \qquad \forall i \in I, \forall k \in \Gamma_i, t \in T \setminus \{0, t_{\max}\} \tag{3.34}$$

$$\sum_{e=0}^{t-1} u_{ik}^{e} - t u_{ik}^{t-1} + t u_{ik}^{t} \leq t \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \setminus \{0\} \tag{3.35}$$

$$\sum_{i \in \Omega_k} \Delta_{ik}^{t} \leq \pi \qquad \forall k \in K, \forall t \in T \tag{3.36}$$

$$\sum_{i \in \Phi_m} \sum_{k \in \Gamma_i} \Delta_{ik}^{t} \leq \pi \qquad \forall m \in M, \forall t \in T \tag{3.37}$$

$$u_{ik}^{t} + u_{jk}^{t} + u_{ik}^{t+1} + u_{jk}^{t+1} \leq 3 \qquad \begin{array}{r} \forall k \in K, \forall i, j \in \Omega_k, i \neq j, \\ \forall t \in T \setminus \{t_{\max}\} \end{array} \tag{3.38}$$

$$\sum_{k \in \Gamma_i} u_{ik}^{t} + \sum_{k' \in \Gamma_j} u_{jk'}^{t} + \sum_{k \in \Gamma_i} u_{ik}^{t+1} + \sum_{k' \in \Gamma_j} u_{jk'}^{t+1} \leq 3 \qquad \begin{array}{r} \forall m \in M, \\ \forall i, j \in \Phi_m, i \neq j, \\ \forall t \in T \setminus \{t_{\max}\} \end{array} \tag{3.39}$$

$$s_{ik}^{t} \geq u_{ik}^{t} + u_{ik}^{t-1} - 1 \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \setminus \{0\} \tag{3.40}$$

$$f_{ik}^{t} \geq u_{ik}^{t} + u_{ik}^{t+1} - 1 \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \setminus \{t_{\max}\} \tag{3.41}$$

$$c_{ik}^{t} \leq \Delta_{ik}^{t} + \pi(1 - s_{ik}^{t}) \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \tag{3.42}$$

$$c_{ik}^{t} \geq \pi f_{ik}^{t} \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \tag{3.43}$$

$$c_{ik}^{t} \leq \pi u_{ik}^{t} \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \tag{3.44}$$

$$c_{ik}^{t} \geq \Delta_{ik}^{t} \qquad \forall k \in K, \forall i \in \Omega_k, \forall t \in T \tag{3.45}$$

$$c_{jk}^{t} \geq c_{ik}^{t} + \Delta_{jk}^{t} - \pi(1 - \psi_{ij}^{t}) \qquad \forall k \in K, \forall i, j \in \Omega_k, i \neq j, \forall t \in T \tag{3.46}$$

$$\sum_{k \in \Gamma_j} c_{jk}^{t} \geq \sum_{k \in \Gamma_i} c_{ik}^{t} + \sum_{k \in \Gamma_j} \Delta_{jk}^{t} - \pi(1 - \mu_{ij}^{t}) \qquad \begin{array}{r} \forall m \in M, \\ \forall i, j \in \Phi_m, i \neq j \end{array} \tag{3.47}$$

$$\psi_{ij}^{t} + \psi_{ji}^{t} \geq u_{ik}^{t} + u_{jk}^{t} - 1 \qquad \forall k \in K, \forall i, j \in \Omega_k, \forall t \in T \tag{3.48}$$

$$\mu_{ij}^{t} + \mu_{ji}^{t} \geq \sum_{k \in \Gamma_i} u_{ik}^{t} + \sum_{k \in \Gamma_j} u_{jk}^{t} - 1 \qquad \forall m \in M, \forall i, j \in \Phi_m, \forall t \in T \tag{3.49}$$

$$\sum_{k \in \Gamma_i} \sum_{\substack{t \in T \\ t \leq t_i^{\alpha} - 1 \\ t \geq t_i^{\beta} + 1 \\ t \geq t_k^{h} + 1}} u_{ik}^{t} = 0 \qquad \forall i \in I \tag{3.50}$$

$$\Delta_{ik}^{t_i^{\alpha}} \leq (\pi - \delta_i^{\alpha}) \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.51}$$

$$\Delta_{ik}^{t_i^{\beta}} \leq \delta_i^{\beta} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.52}$$

$$\Delta_{ik}^{t_k^{h}} \leq \delta_k^{h} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.53}$$

$$c_{ik}^{t_i^{\alpha}} \geq \begin{cases} (\delta_i^{\alpha} + p_i) u_{ik}^{t_i^{\alpha}} & \text{se } (\delta_i^{\alpha} + p_i) \leq \pi \\ \pi u_{ik}^{t_i^{\alpha}} & \text{caso contrário} \end{cases} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.54}$$

$$c_{ik}^{t_i^\beta} \leq \delta_i^\beta u_{ik}^{t_i^\beta} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.55}$$

$$c_{ik}^{t_k^h} \leq \delta_k^h u_{ik}^{t_k^h} \qquad \forall i \in I, \forall k \in \Gamma_i \tag{3.56}$$

$$0 \leq \Delta_{ik}^t \leq \pi, 0 \leq c_{ik}^t \leq \pi \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \tag{3.57}$$

$$u_{ik}^t \in \{0,1\}, s_{ik}^t \in \{0,1\}, f_{ik}^t \in \{0,1\} \qquad \forall i \in I, \forall k \in \Gamma_i, \forall t \in T \tag{3.58}$$

$$\psi_{ij}^t \in \{0,1\} \qquad \forall i \in I, \forall j \in I, i \neq j, \forall t \in T \tag{3.59}$$

$$\mu_{ij}^t \in \{0,1\} \qquad \forall i \in I, \forall j \in I, i \neq j, e_i = e_j, \forall t \in T \tag{3.60}$$

A função objetivo (3.26) é igual à função objetivo das formulações NET e TI. As restrições (3.28) garantem que o somatório dos tempos de execução nos períodos aos quais a manutenção i foi alocada seja igual ao seu tempo de execução. As restrições (3.29) e (3.30) definem um limite superior e um limite inferior, respectivamente, para o número de períodos aos quais cada manutenção pode ser alocada. As restrições (3.31) impõem que a variável Δ_{ik}^t seja igual a zero, se a manutenção i não estiver alocada ao período t e equipe k. As restrições (3.32) e (3.33) garantem que se uma manutenção for alocada para ser executada por uma equipe em qualquer período, esta manutenção deve ser alocada a esta equipe. As restrições (3.34) garantem que, caso uma manutenção esteja alocada em três ou mais períodos, o valor das variáveis Δ_{ik}^t nos períodos intermediários devem ser iguais a π, para garantir a não preempção. As restrições (3.35) garantem que os períodos serão atribuídos de maneira consecutiva para as manutenções executadas. Estas restrições são baseadas na formulação proposta em Barros et al. (2011). As restrições (3.36) e (3.37) garantem que o tempo total de execução das manutenções em um mesmo período e na mesma equipe ou máquina, respectivamente, seja menor ou igual a π. As restrições (3.38) e (3.39) complementam estas últimas restrições, garantindo que manutenções associadas à mesma equipe e à mesma máquina, respectivamente, não sejam executadas simultaneamente quando alocadas em períodos consecutivos.

As restrições (3.40)-(3.49) determinam o sequenciamento e a programação das manutenções nos períodos. As restrições (3.40) garantem que s_{ik}^t será igual a 1 se a manutenção i estiver alocada aos períodos consecutivos $t-1$ e t. As restrições (3.41) garantem que f_{ik}^t será igual a 1 se a manutenção i estiver alocada aos períodos t e $t+1$. As restrições (3.42) impõem que o tempo de conclusão c_{ik}^t de uma certa manutenção no período t deve ser igual a Δ_{ik}^t se a execução da manutenção iniciar no período $t-1$ ou antes. Assim, estas restrições forçam que a primeira manutenção a ser executada no período seja a manutenção inicializada em períodos anteriores. As restrições (3.43) impõem, para uma manutenção que é executada tanto no período t quanto no período seguinte, que o seu tempo de conclusão no período t deve ser igual a π, ou seja, esta manutenção é programada para ser executada no final do período. As restrições (3.44) garantem que o tempo de conclusão de uma manutenção em uma equipe no período t será igual a 0, caso a manutenção não esteja alocada a esta equipe neste período. Por outro lado, as restrições (3.45) impõem que o tempo de conclusão de uma manutenção em uma equipe em um determinado período seja maior ou igual ao tempo

de execução desta manutenção nesta equipe neste período. As restrições (3.46) e (3.47) garantem que o tempo de conclusão de uma manutenção seja maior que o tempo de conclusão da manutenção que a precedeu na mesma equipe ou máquina, respectivamente, mais o seu tempo de processamento. Assim, essas restrições garantem que duas manutenções alocadas à mesma equipe ou máquina não sejam executadas simultaneamente. As restrições (3.48) e (3.49) garantem que, caso duas manutenções sejam alocadas à mesma equipe ou máquina, respectivamente, no período t, será definida uma sequência de execução, ou seja, uma delas deve preceder a outra.

As restrições (3.50)-(3.56) são responsáveis por garantir que as manutenções sejam executadas dentro das suas respectivas janelas de tempo e do tempo de serviço das equipes de trabalho. As restrições (3.50) garantem que não haverá execução das manutenções nos períodos que não pertencem à janela de tempo e ao intervalo de serviço das equipes associadas a cada manutenção. As restrições (3.51)-(3.53) definem um limite para as variáveis $\Delta_{ik}^{t_i^\alpha}$, $\Delta_{ik}^{t_i^\beta}$ e $\Delta_{ik}^{t_i^h}$ nos períodos t_i^α, t_i^β e t_k^h. A restrição (3.51) considera que, como o valor de δ_i^α representa o tempo entre o início do período e o momento em que se inicia a janela, o tempo disponível para a manutenção i no período t_i^α é a duração total do período menos o tempo que inicia a janela. As restrições (3.54)-(3.56) estão associadas ao tempo de conclusão da execução que dura Δ_{ik}^t nos períodos t_i^α, t_i^β e t_i^h. As restrições (3.54) apresentam um limite superior para o tempo de conclusão de uma manutenção que começa a ser executada no início de sua janela de tempo. As restrições (3.55) e (3.56) garantem que o tempo de conclusão da manutenção deve ser menor ou igual ao final da janela da manutenção e ao final do intervalo de serviço da equipe, respectivamente. Por fim, as restrições (3.57)-(3.60) definem os domínios das variáveis de decisão.

3.6 Análise e comparação das dimensões dos modelos

Para efeito de comparação, a Tabela 3.3 fornece a quantidade de restrições e de variáveis das formulações apresentadas. Pode-se notar que os termos que mais afetam o número de restrições das formulações NET e TIP são, respectivamente, $|I|^2.|K|^2$ e $|I|^2.|T|.(|M|+|K|)$. Para o número de variáveis, os termos que exercem maior influência para as formulações NET e TIP são, respectivamente, $|I|^2(|K|+1)$ e $|T|.|I|(|I|+|K|)$. Diferentemente das formulações NET e TIP, a formulação TI depende de um número pseudo-polinomial de variáveis e restrições. No entanto, o número de variáveis e restrições depende diretamente do número de períodos, o qual pode ser um número relativamente muito grande.

Ao comparar a formulação TI com a TIP, existe uma diferença no número total de períodos. Essa diferença está vinculada ao parâmetro π, que define a duração do período e em quantos períodos o horizonte de planejamento é dividido. Como o número de períodos é inversamente proporcional à duração dos períodos e o parâmetro π pode ter qualquer valor,

Tabela 3.3: Dimensões das formulações.

Formulação	#restrições	#variáveis
NET	$\|K\|.\|I\|.(\|K\|.\|I\| - \|K\| + 6) + 2\|K\| + \|I\|$	$\|K\|.(\|I\|^2 + \|I\| + 1) + \|I\|^2 - \|I\|$
TI	$\|K\|.(2\|I\| + \|T\|) + \|T\|.\|M\| + 2\|I\|$	$\|K\|.(\|I\|.\|T\| + \|I\| + 1)$
TIP	$\|I\|.(\|I\| - 1).(\|M\| + \|K\|).(3\|T\| - 1)$ $+\|I\|.\|K\|.(10\|T\| + 4) + \|T\|.(\|M\| + \|K\|) + \|I\|.(\|T\| - 3)$	$\|T\|.(\|I\|^2 - \|I\|) + \|K\|.(5\|I\|.\|T\| - \|I\| + 1)$

a formulação TIP pode ser descrita com um número menor de períodos do que a formulação TI. Por exemplo, em um caso em que as manutenções possuem um tempo de execução fracionário, a duração do período na TI pode ter que ser igual a 10% hora, enquanto que, na formulação TIP, pode ser qualquer valor maior que zero, incluindo valores inteiros. Como π está associado a $|T|$, esse parâmetro afeta fortemente o número de variáveis e de restrições, ou seja, a dimensão da formulação. Uma análise empírica da influência do valor de π na formulação TIP é apresentada na Seção 3.8.2.1.

A alocação simultânea das manutenções às equipes e às máquinas e a restrição de que manutenções associados à mesma máquina não podem ser executadas simultaneamente são características importantes do LTPMOSP e que influenciam as formulações. A condição citada é garantida na formulação TI através das restrições (3.24). Elas são responsáveis pela construção da sequência de execução das manutenções nas máquinas, sem que tenha sobreposição de manutenções. As restrições (3.24) são semelhantes as (3.23), pois da mesma forma que se tem uma programação nas equipes, tem-se nas máquinas. Como ambas as restrições estão associadas com períodos, é gerado um grande número de restrições quando a quantidade de períodos é grande. Por outro lado, a formulação TIP utiliza um conjunto de variáveis e restrições para garantir que manutenções associados à mesma máquina não sejam executados ao mesmo tempo, tornando a formulação mais extensa. As variáveis de precedência e de conclusão, e as restrições que utilizam estas variáveis, foram adicionadas à formulação porque apenas as restrições (3.36) e (3.38) não são suficientes para garantir a não sobreposição.

3.7 Métodos de solução propostos baseados no algoritmo GRASP

Esta seção introduz dois métodos de solução baseados no algoritmo GRASP, propostos para solucionar o LTPMOSP. O primeiro, nomeado como GRASP, é uma implementação direta dessa meta-heurística. O segundo, chamado GRASP+Formulation, é um método exato hibrido que usa o algoritmo GRASP proposto para gerar uma solução inicial para o *solver*. Esta seção está organizada da seguinte forma. A Subseção 3.7.1 descreve o método GRASP, enquanto, a Subseção 3.7.2 apresenta o método exato híbrido.

3.7.1 Método heurístico GRASP

Esta seção está organizada da seguinte forma: a Subseção 3.7.1.1 mostra como uma solução para LTPMOSP é representada. A Subseção 3.7.1.2 descreve o algoritmo para gerar uma solução, atribuir manuteções às equipes e definir o tempo de conclusão das manutenções. A Subseção 3.7 apresenta o algoritmo GRASP proposto. A Subseção 3.7.1.4 detalha o procedimento de busca local deste método GRASP.

3.7.1.1 Representação da solução

Para representar uma solução do LTPMOSP, é utilizado um vetor $s = \langle s_1, \ldots, s_{|I|} \rangle$ com $|I|$ posições. Nesta representação, cada posição s_i do vetor indica uma manutenção. A sequência em que as manutenções são alocadas nas posições do vetor solução s é definida pelo algoritmo GRASP apresentado na Subseção 3.7.1.3. Como em Aquino et al. (2018a), esta pesquisa optou por utilizar esta representação indireta da solução, pois ela simplifica a construção de soluções e a exploração do espaço de soluções viáveis utilizando operadores de vizinhança mais simples em relação à representação direta, em que há uma lista de manutenções para cada equipe e máquina. De posse de uma solução s neste formato, uma solução para o LTPMOSP é obtida através de um algoritmo de posicionamento de manutenções descrito na Seção 3.7.1.2.

3.7.1.2 Algoritmo de posicionamento de manutenções

Dado o vetor s representando uma sequência de manutenções, um subproblema de posicionamento pode ser usado para gerar e decodificar uma solução. Esse subproblema, que tem o mesmo objetivo do problema original, concentra-se em determinar, para um vetor s, (i) as equipes ativadas; (ii) as manutenções programadas; (iii) a atribuição das manutenções programadas a uma equipe ativada; e (iv) a programação das manutenções dentro de sua janela. Resolver esse subproblema é bastante desafiador, pois também é um problema NP-difícil, de acordo com a Proposição 1 no Apêndice A. Portanto, devido à complexidade deste subproblema, um procedimento heurístico, ou seja, o algoritmo de posicionamento de manutenções (PM), foi proposto para abordá-lo para um vetor s. O Algoritmo 2 apresenta seu pseudocódigo.

O Algoritmo 2 utiliza uma lista de prioridades para cada manutenção i, denotada por $\Theta_i = \left(\theta_i^1, \ldots, \theta_i^{|\Gamma_i|}\right)$. Essa lista possui $|\Gamma_i|$ posições, cada uma representando uma equipe. Cada lista Θ_i possui apenas equipes com a habilidade para executar a manutenção i. Essa lista de prioridades é baseada na ordenação decrescente do valor h_k para $k \in \Gamma_i$. Assim, as equipes com o maior valor de h_k são as primeiras da lista.

O Algoritmo PM assume que o valor inicial da função objetivo z é igual a zero (linha 1). Todas as equipes começam desabilitadas ($y_k = \mathit{false}$) nas linhas 2 a 4 e sem manutenções atribuídas. Na linha 5, um processo iterativo é iniciado, o qual, em cada iteração, tenta atribuir exatamente uma manutenção (s_i). Na linha 6, a variável booleana *manutencaoAtribuida*,

Algoritmo 2: PM

Entradas: solução s
Saídas : valor de função objetivo (z)

1 $z \leftarrow 0$
2 **for** *cada equipe* $k \in K$ *de* $k \leftarrow 1$ **to** $|K|$ **do**
3 $\quad y_k \leftarrow false$
4 **end**
5 **for** *cada manutenção* s_i *de* $i \leftarrow 1$ **to** $|I|$ **do**
6 $\quad manutencaoAtribuida \leftarrow false$
7 $\quad$ **for** *cada equipe* $\theta^l_{s_i}$ *de* $l \leftarrow 1$ **to** $|\Gamma_{s_i}|$ **do**
8 $\quad\quad \eta = \min(\beta_{s_i}, h_{\theta^l_{s_i}})$
9 $\quad\quad manutencaoAtribuida \leftarrow encontraIntervalo(s_i, \theta^l_{s_i}, \alpha_{s_i}, \eta)$
10 $\quad\quad$ **if** $manutencaoAtribuida = true$ **then**
11 $\quad\quad\quad$ Atribui a manutenção s_i à equipe $\theta^l_{s_i}$
12 $\quad\quad\quad$ Atribui a manutenção s_i à respectiva máquina
13 $\quad\quad\quad$ **if** $y_k = false$ **then**
14 $\quad\quad\quad\quad y_k \leftarrow true$
15 $\quad\quad\quad\quad z \leftarrow z + 1$
16 $\quad\quad\quad$ **end**
17 $\quad\quad\quad$ retorna para a linha 5. `// Procede para a próxima manutenção do vetor` s.
18 $\quad\quad$ **end**
19 $\quad$ **end**
20 $\quad$ **if** $manutencaoAtribuida = false$ **then**
21 $\quad\quad z \leftarrow z + w_{s_i}$
22 $\quad$ **end**
23 **end**
24 **return** z

que controla se a manutenção é atribuída (*true*) ou não (*false*), recebe *false*. Na linha 7, um processo iterativo é iniciado, o qual, em cada iteração, tenta atribuir a manutenção s_i a uma equipe da lista Θ_{s_i}. Este processo começa com a equipe $\theta^1_{s_i}$. Na linha 8, o valor do parâmetro $\eta = \min(\beta_{s_i}, h_{\theta^l_{s_i}})$ é determinado. Este parâmetro define a janela de tempo resultante $[\alpha_{s_i}, \eta]$ em que a manutenção s_i pode ser atribuída à equipe $\theta^l_{s_i}$.

Na linha 9, a função $encontraIntervalo(\cdot)$ é executada. Ela é usada para encontrar um intervalo de tempo p_{s_i} disponível na equipe $\theta^l_{s_i}$ para atribuir a manutenção s_i a ela. Este intervalo de tempo deve estar dentro da janela de tempo resultante $[\alpha_{s_i}, \eta]$ e não se sobrepor a nenhuma manutenção atribuída à equipe $\theta^l_{s_i}$ e à máquina associada à manutenção s_i. O Algoritmo 3 apresenta o pseudocódigo da função $encontraIntervalo(\cdot)$.

Se a função $encontraIntervalo(\cdot)$ retornar $manutencaoAtribuida = true$, a manutenção s_i é atribuída ao mesmo instante do intervalo disponível encontrado na função para a equipe e a máquina associadas à manutenção (linhas 11 e 12). Além disso, se a equipe do intervalo disponível foi desabilitada ($y_{\theta^l_{s_i}} = false$) e a manutenção s_i é a primeira a ser atribuída, então a equipe é considerada ativada e $y_{\theta^l_{s_i}}$ é definido como *true* (linha 14). Consequentemente,

o valor de z deve ser incrementado em uma unidade (linha 15). Após a manutenção s_i ter sido atribuída, o processo iterativo prossegue para a próxima manutenção do vetor s, ou seja, a manutenção s_{i+1}.

Por outro lado, se a função $encontraIntervalo(\cdot)$ retorna $manutencaoAtribuida = false$ (linha 20), a manutenção s_i não é programada. Então, uma penalidade para a respectiva manutenção é adicionada ao valor da função objetivo (linha 21), e o algoritmo PM passa para a próxima manutenção do vetor s.

Após o processo iterativo tentar atribuir todas as manutenções do vetor s, o algoritmo PM é finalizado. Cada execução completa do algoritmo gera uma solução viável para o LTPMOSP.

Portanto, a decodificação do vetor s para encontrar uma solução para o problema é realizada nas seguintes linhas do algoritmo: (i) as equipes ativadas são encontradas na linha 14; (ii) as manutenções programadas são fornecidos na linha 9; (iii) a atribuição das manutenções programadas a uma equipe é fornecida na linha 11; e (iv) a programação das manutenções dentro de sua janela de tempo designada é definida na função $encontraIntervalo(\cdot)$, chamada na linha 9. Uma descrição do procedimento para definir o horário de início e término da programação para cada manutenção, na função $encontraIntervalo(\cdot)$, é apresentada no Algoritmo 3.

O Algoritmo 3 começa com algumas variáveis auxiliares recebendo $False$ (linhas 1 a 3). Na linha 4, a variável *tempoFinal*, usada para armazenar o tempo final do intervalo disponível encontrado e limitar a busca, recebe η. Ela recebe esse parâmetro para evitar a busca por um intervalo após a janela final da manutenção s_i. O *loop* principal se inicia na linha 6. Este procedimento iterativo é responsável por buscar e encontrar um intervalo disponível dentro da janela de tempo resultante da equipe e da máquina. Na linha 7, uma busca é realizada para encontrar um intervalo disponível na equipe $\theta^l_{s_i}$. Este intervalo não pode se sobrepor a nenhuma manutenção atribuída à equipe $\theta^l_{s_i}$. Toda busca começa no instante *tempoFinal* e continua em direção ao início da janela resultante. A busca termina após encontrar o primeiro intervalo disponível. Na linha 8, o valor da variável *tempoFinal* é atualizado. Se o início do intervalo ($tempoFinal - p_{s_i}$) ocorrer após o início da janela resultante, a variável *equipeIntervalo* receberá o valor *true*, indicando que o intervalo encontrado na equipe $\theta^l_{s_i}$ está disponível. Caso contrário, a variável *buscaFim* receberá o valor *true*, e o procedimento iterativo será encerrado pois nenhum intervalo disponível foi encontrado dentro da janela de tempo da manutenção s_i.

Se $equipeIntervalo = true$, o algoritmo avança para a próxima etapa, que consiste em verificar se, no intervalo disponível da equipe $\theta^l_{s_i}$, há alguma manutenção atribuída à máquina da manutenção s_i (linha 15). Se não houver conflito, um intervalo disponível foi encontrado nas janelas de tempo da equipe $\theta^l_{s_i}$ e da máquina associada à manutenção s_i, e, então, *manutencaoAtribuida* recebe o valor *true* (linha 17), e a busca é encerrada (linha 18). Por outro lado, se houver conflito, o intervalo encontrado não está disponível para a conclusão da manutenção na máquina, então *equipeIntervalo* recebe o valor *false*.

Algoritmo 3: *encontraIntervalo(.)*

Entradas: manutenção s_i, equipe $\theta^l_{s_i}$, janela de tempo resultante $[\alpha_{s_i}, \eta]$
Saídas : *true* ou *false* se manutenção s_i foi atribuída a equipe $\theta^l_{s_i}$. Se *true* é indicado o instante de tempo de execução da respectiva manutenção

1 *manutencaoAtribuida* ← *false* `// Variável para indicar se a manutenção` s_i `pode ser atribuída à equipe` $\theta^l_{s_i}$ `e sua máquina`
2 *equipeIntervalo* ← *false* `// Variável para indicar se há um intervalo disponível na equipe` $\theta^l_{s_i}$
3 *maquinaIntervalo* ← *false* `// Variável para testar se há um intervalo disponível na máquina da manutenção` s_i
4 *tempoFinal* ← η `// Variável para armazenar o tempo final do intervalo encontrado`
5 *buscaFim* ← *false*
6 **while** *(buscaFim == false)* **do**
7 Procure o primeiro intervalo disponível antes do instante *tempoFinal* na equipe $\theta^l_{s_i}$
8 *tempoFinal* ← tempo final do intervalo encontrado
9 **if** *((tempoFinal*$-p_{s_i}$*)* $\geq \alpha_{s_i}$*)* **then**
10 *equipeIntervalo* ← *true*
11 **else**
12 *buscaFim* ← *true*
13 **end**
14 **if** *(equipeIntervalo == true)* **then**
15 Verifique se, no intervalo disponível da equipe $\theta^l_{s_i}$, há alguma manutenção atribuída à máquina da mantenção s_i
16 **if** *(não há conflito)* **then**
17 *manutencaoAtribuida* ← *true*
18 *buscaFim* ← *true*
19 **else**
20 *equipeIntervalo* ← *false*
21 **end**
22 **end**
23 **if** *(buscaFim == false)* **then**
24 Procure o primeiro intervalo disponível antes do instante *tempoFinal* na máquina da manutenção s_i
25 *tempoFinal* ← tempo final do intervalo encontrado
26 **if** *((tempoFinal*$-p_{s_i}$*)* $\geq \alpha_{s_i}$*)* **then**
27 *maquinaIntervalo* ← *true*
28 **else**
29 *buscaFim* ← *true*
30 **end**
31 **if** *(maquinaIntervalo == true)* **then**
32 Verifique se há alguma manutenção atribuída à equipe $\theta^l_{s_i}$ no intervalo disponível da máquina da manutenção s_i
33 **if** *(não há conflito)* **then**
34 *manutencaoAtribuida* ← *true*
35 *buscaFim* ← *true*
36
37 **else**
38 *maquinaIntervalo* ← *false*
39 **end**
40 **end**
41 **end**
42 **end**
43 **return** *manutencaoAtribuida*

Nesse caso, a próxima etapa do algoritmo visa buscar um intervalo diferente do encontrado anteriormente.

Na linha 24, um intervalo disponível na máquina do manutenção s_i é pesquisado. Diferentemente da pesquisa anterior, esta considera apenas as manutenções atribuídas à máquina. No entanto, similarmente à linha 7, a pesquisa é feita do final para o início da janela de tempo resultante, começando no instante *tempoFinal*. Esta pesquisa termina após encontrar o primeiro intervalo antes do instante *tempoFinal*. Como este novo intervalo

começa antes do anterior, ele pode ter um tempo de conclusão antes das manutenções com os quais o intervalo anterior entrou em conflito. Na linha 25, o valor da variável *tempoFinal* é atualizado com o novo intervalo disponível encontrado. Se o início do intervalo disponível ($tempoFinal - p_{s_i}$) começar após o início da janela de tempo resultante, então *maquinaIntervalo* recebe o valor *true*, indicando que o intervalo encontrado está disponível na máquina. No entanto, é necessário verificar se ele também está disponível para a equipe. Caso contrário, *buscaFim* recebe o valor *true*, e o procedimento iterativo termina.

Se a variável *equipeIntervalo* = *true*, na linha 33, verifica-se se, no intervalo encontrado na máquina da manutenção s_i, há alguma manutenção atribuída à equipe $\theta^l_{s_i}$. Se não houver conflito, um intervalo disponível foi encontrado na equipe $\theta^l_{s_i}$ e na máquina, então *manutencaoAtribuida* recebe o valor *true* (linha 34), e a busca é concluída. Caso contrário, não há intervalo disponível na equipe $\theta^l_{s_i}$ para executar a manutenção s_i, então a variável *machineInterval* recebe o valor *false*. O próximo passo do algoritmo é iniciar uma nova busca por um intervalo diferente daquele encontrado na linha 26. Assim, o algoritmo retorna à linha 7 e repete o procedimento de busca por um intervalo disponível na equipe $\theta^l_{s_i}$ e a verificação de um conflito na máquina da manutenção s_i. Portanto, o procedimento iterativo de busca e verificação de um intervalo disponível é repetido até que nenhum intervalo disponível seja encontrado dentro da janela de tempo resultante $[\alpha_{s_i}, \eta]$.

A complexidade dos algoritmos PM e *encontraIntervalo* é investigada. Em Algorithm 2, a linha 2 é executada $|K|$ vezes. Por sua vez, no loop de repetição das linhas 6 a 23, a linha 6 é executada $|I|$ vezes, enquanto a linha 7 é executada $|\Gamma_{s_i}|$ vezes, onde Γ_{s_i} é o subconjunto de equipes de trabalho que possui a habilidade necessária para executar a manutenção s_i. Como $\Gamma_i \subseteq K$, então, no pior caso, temos $|\Gamma_{s_i}| = |K|$, ou seja, a linha 7 é executada $|K|$ vezes. Por outro lado, na função *encontraIntervalo*, descrita no Algoritmo 2, a janela de tempo de cada manutenção s_i é percorrida do fim para o início, ou seja, no pior caso, essa operação consome $|\beta_{s_i} - \alpha_{s_i}| \leq |H|$ unidades de tempo $\forall i \in I$, onde H é o horizonte de planejamento. Portanto, no pior caso, essa operação requer $|I| \times |H|$ unidades de tempo para cobrir as janelas de tempo de todas as manutenções. Como é necessário verificar se a máquina associada à manutenção está disponível para cada instante de tempo de cada manutenção, essa verificação leva uma unidade de tempo. Portanto, a função *encontraIntervalo* tem complexidade de tempo $O(|I| \times |H|)$. Então, o Algoritmo 2 tem complexidade de tempo $O(|I|^2 \times |K| \times |H|)$.

Como mencionado anteriormente, esta tese utiliza a representação de solução introduzida por Aquino et al. (2018a,b, 2019). No entanto, ela propõe um algoritmo diferente de posicionamento de manutenções para construir uma solução viável e avaliá-la. Este novo algoritmo busca atribuir as manutenções no final de suas janelas de tempo, enquanto aqueles autores as atribuem no início. Esse foi utilizado em Coelho et al. (2026). Além disso, o algoritmo proposto cria uma lista de equipes prioritárias para executar cada manutenção, enquanto o algoritmo dos artigos citados não utiliza essa lista de prioridades. Outra diferença está na construção do vetor de solução s. Enquanto nos artigos citados este vetor é

previamente definido de acordo com as penalidades das manutenções, na presente tese, por outro lado, um algoritmo GRASP, apresentado na subseção a seguir, o determina.

3.7.1.3 Algoritmo GRASP

Diferentemente da versão padrão de um algoritmo GRASP, revisada na Seção 2.5.1, o algoritmo proposto no presente trabalho não executa a fase de busca local em cada iteração. A fase de busca local é executada apenas uma vez, com a melhor solução encontrada no processo iterativo de construção de novas soluções. Com esta estratégia, é possível explorar melhor o Algoritmo PM e obter um tempo relativamente reduzido de execução do algoritmo meta-heurístico. Esta estratégia é adotada, pois a fase de busca local é muito custosa computacionalmente e executá-la em todas as iterações elevaria muito o tempo computacional do Algoritmo GRASP, principalmente nas instâncias de maior escala. Além disso, o Algoritmo PM é capaz de gerar boas soluções. O Algoritmo 4 mostra o pseudocódigo do algoritmo GRASP proposto neste trabalho.

Algoritmo 4: *GRASP*

Entrada: $GRASPmax$, α, $RDmax$
Saídas : solução s^*

```
1  z* ⟵ ∞
2  s* ⟵ ∅
3  s ⟵ ∅
4  it ⟵ 0
5  while (it ≤ GRASPmax) do
6  |   s ⟵ faseConstrucao(α)
7  |   z ⟵ PM(s)
8  |   if (z < z*) then
9  |   |   s* ⟵ s
10 |   |   z* ⟵ z
11 |   |   it ⟵ 0
12 |   end
13 |   else
14 |   |   it ⟶ it + 1
15 |   end
16 end
17 if instância é pequena then
18 |   s⋆ ⟵ buscaLocal-BI(s⋆)
19 end
20 else
21 |   s⋆ ⟵ buscaLocal-RD(s⋆, RDmax)
22 end
23 return s*
```

Nas quatro primeiras linhas do algoritmo GRASP são definidos os valores iniciais das variáveis que armazenam o valor da função objetivo ($z^\star$), a melhor solução ($s^\star$), a solução atual (s) e o contador de iterações (it). Em seguida, um procedimento iterativo é iniciado

(linhas 5 a 17), que é interrompido quando o número máximo de iterações ($GRASPmax$) sem melhoria na melhor solução é atingido. Na linha 6, a função $faseConstrucao(\alpha)$ é executada. Esta função atribui todas as manutenções disponíveis a uma posição no vetor de soluções s. A descrição do procedimento da construção do vetor s através da função é apresentada no Algoritmo 5. Com o vetor s definido, o algoritmo PM é executado para obter uma solução para o problema (linha 7). Se o valor da função objetivo z da solução fornecida pelo algoritmo PM for melhor que z^* (linha 8), ela é aceita como a melhor solução até o momento, e o contador it retorna a zero; caso contrário, o contador it é incrementado em uma unidade. Após $GRASPMax$ iterações sem melhorar a melhor solução, o *loop* principal é concluído e o algoritmo aplica o procedimento de busca local descrito na Subseção 3.7.1.4.

Algoritmo 5: $faseConstrucao(\alpha)$

Entradas: α
Saídas : vetor s

```
1  s ⟵ ∅
2  Crie uma LC
3  Crie uma LRC com as α manutenções de maiores valores de penalidades da LC
4  j ⟵ 0
5  while j < |I| do
6  |  Selecione aleatoriamente uma manutenção da LRC
7  |  Adicione a manutenção selecionada na posição m da solução s
8  |  Remova a manutenção selecionada da LRC
9  |  Insira na LRC a manutenção de maior penalidade ainda não inserida
10 |  j = j + 1
11 end
12 return s
```

O Algoritmo 5 inicia com o vetor s sem nenhuma manutenção (linha 1). Na linha 2 é criada uma lista de candidatos (LC). Ela é uma lista de todas as manutenções classificadas, em ordem decrescente de acordo com o valor da penalidade. Em seguida, na linha 3 é criada uma lista de candidatos restrita (LRC) com α posições usando a LC. A LRC é inicialmente composta pelo número limite α de manutenções com os maiores valores de penalidade em LC. Após a construção da LRC, é iniciado um processo iterativo que busca posicionar todas as manutenções disponíveis no vetor s. Ele ocorre entre as linhas 5 e 11. Este processo tem as seguintes etapas: selecionar aleatoriamente uma manutenção da LRC 6; adicionar a manutenção selecionada à solução s 7; remover a manutenção selecionada da LRC 8; e inserir na LRC a manutenção com a maior penalidade ainda não inserida 9. Este procedimento é repetido até que todas as manutenções disponíveis sejam atribuídas.

Para ilustrar o funcionamento da função $faseConstrucao(\alpha)$, considere $\alpha = 3$ e o exemplo da Figura 3.1. Com esses dados de entrada, a função inicia com LC = {5, 2, 1, 7, 6, 4, 3} e LRC = {5, 2, 1}. Inicia-se, então, um procedimento iterativo, que segue os seguintes passos. Primeiramente, na Etapa (i), seleciona-se aleatoriamente uma manutenção da LRC, ou seja, a manutenção 2. A Etapa (ii) adiciona essa manutenção à primeira posição do

vetor s, por exemplo, $s = \langle 2 \rangle$. Na Etapa (iii), a manutenção 2 é removida da LRC e a manutenção 7 é inserida na LRC na Etapa (iv), resultando em LRC = {5, 1, 7}. Inicia-se, então, uma nova iteração do procedimento. Supondo que a manutenção 5 seja selecionada aleatoriamente, ela é adicionada à segunda posição do vetor s, ou seja, $s = \langle 2, 5 \rangle$. Em seguida, o procedimento remove a manutenção 5 da LRC e adiciona outra manutenção ainda não analisada à mesma lista. Este procedimento é repetido até que todas as manutenções no vetor s tenham sido alocadas.

3.7.1.4 Busca local

De acordo com a Subseção 3.8.1, este capítulo trata de instâncias de pequena, média e grande escala. Esses conjuntos de instâncias, no entanto, apresentam variações significativas no número de manutenções, equipes de trabalho e máquinas, conforme é mostrado na Tabela 3.4. Esse fator aumenta o custo computacional do procedimento de busca local utilizado e leva à adoção do formato apresentado a seguir para mitigar seu impacto na análise de problemas.

A fase de busca local é implementada por meio do método *best improvement* para instâncias de pequena escala e do método de descida randômica para instâncias de média e grande escala. Esses dois métodos de busca local são descritos a seguir.

O método *best improvement* consiste em partir de qualquer solução inicial, analisar todos os seus vizinhos possíveis a cada passo e avançar apenas para o vizinho que apresentar o melhor valor da função objetivo. O algoritmo para quando um ótimo local é encontrado. O movimento de troca é usado para explorar o espaço de soluções. Este movimento envolve a troca da manutenção s_j na posição j pela manutenção s_r na posição r. A função *buscaLocal-BI* do algoritmo GRASP que implementa o método *best improvement* tem como entrada apenas a solução s.

Como analisar todos os vizinhos gerados por instâncias de média, grande e escala real é impraticável, implementa-se o método de busca local com descida randômica. Este método consiste em analisar um vizinho selecionado aleatoriamente e aceitar este vizinho se ele tiver um valor de função objetivo estritamente menor que a melhor solução obtida até então. Caso contrário, a melhor solução permanece inalterada e outro vizinho é gerado aleatoriamente. Este procedimento é interrompido após um número máximo de iterações sem melhoria na solução, definido pelo parâmetro *RDmax*. Para gerar um vizinho no método de descida randômica, é necessário selecionar aleatoriamente duas posições do vetor s e trocar as posições. A função *buscaLocal-DR* do algoritmo GRASP que implementa o método de descida randômica tem como entrada a solução s e o parâmetro *RDmax*.

Em relação à complexidade, como existem $|I|$ manutenções em uma solução s, a vizinhança de troca envolve $|I| \times (|I| - 1)/2$ movimentos de troca na melhor estratégia de melhoria. Considerando que o algoritmo PM é executado em cada operação de troca e sua complexidade é $O(|I|^2 \times |K| \times |H|)$, então a complexidade dessa estrutura de vizinhança é $O(|I|^4 \times |K| \times |H|)$.

3.7.2 Método exato híbrido

Além das duas novas formulações TI e TIP propostas e do algoritmo heurístico baseado em GRASP, este trabalho também propõe um método exato que combina cada formulação com o algoritmo GRASP. O Algoritmo 6 apresenta o pseudocódigo para este método exato híbrido.

Algoritmo 6: Método exato híbrido

Entradas: $GRASPmax$, α, $RDmax$, F
Saídas : vetor $s^\star$
1 $s \longleftarrow GRASP(GRASPmax, \alpha, RDmax)$
2 $s^\star \longleftarrow$ SolveMIP(s, F)
3 **return** $s^\star$

Este método executa o algoritmo GRASP descrito na Subseção 3.7 para gerar uma solução inicial (s) com um bom valor de função objetivo. A partir dessa solução inicial, um *solver* é executado com a formulação F e retorna a solução ótima, denotada por $s^\star$. Uma vez que o algoritmo GRASP gera uma solução inicial, a heurística do *solver* pode ser desabilitada. Nesse caso, o tempo de execução pode ser reduzido, eliminando-se o tempo gasto pela heurística do *solver* e pela possibilidade de podar mais nós da árvore de ramificação e corte, visto que o *solver* já começa com uma boa solução incumbente.

Este método também pode ser executado com um tempo limite. Nesse caso, o *solver* pode retornar uma solução melhorada, possivelmente ótima. Além disso, como o *solver* fornece a solução final deste método híbrido, ele também fornece um limite inferior, que permite o cálculo do gap entre o valor da função objetivo desta solução e este limite.

3.8 Experimentos computacionais em instâncias do LTPMOSP

Esta seção apresenta os resultados dos experimentos computacionais referentes à formulação NET, de Aquino et al. (2018a), e às formulações TI e TIP e ao algoritmo GRASP propostos neste trabalho. Além disso, os experimentos são comparados com os resultados dos métodos contidas nos trabalhos de Aquino et al. (2018a,b, 2019) e Santos et al. (2022). Os experimentos computacionais foram executados em um computador equipado com um processador Intel i3-10100T 3.00GHz, 16 Gb de RAM e sistema operacional Ubuntu 18.04.6 LTS. Todas as formulações citadas foram implementadas na linguagem `C++` e solucionadas pelo *solver* CPLEX 12.9. O algoritmo GRASP foi implementado usando a linguagem `C++`. Em todas as tabelas apresentadas nesta seção, FN, FTI e FTIP referem-se, respectivamente, ás formulações NET, TI e TIP.

Esta seção está organizada da seguinte forma. A Subseção 3.8.1 descreve as características das instâncias utilizadas nos experimentos computacionais. A Subseção 3.8.2 relata e analisa os resultados dos métodos exatos, enquanto a Subseção 3.8.3 apresenta os resultados do algoritmo heurístico.

3.8.1 Instâncias

Os experimentos computacionais foram realizados com as instâncias de Aquino et al. (2018a), disponíveis em `www.decom.ufop.br/prof/marcone/projects/LTPMSP.html`. Como o número de manutenções nas instâncias varia de 20 a 33.484, elas foram divididas em quatro conjuntos, conforme mostrado na Tabela 3.4. Nessa tabela, a primeira coluna refere-se aos conjuntos de instâncias ($\mathcal{P}$, $\mathcal{M}$, $\mathcal{G}$, $\mathcal{GG}$). A segunda coluna refere-se ao número de instâncias no respectivo conjunto. Portanto, são 139 instâncias. A terceira, a quarta e a quinta colunas descrevem os números mínimo e máximo de manutenções, máquinas e equipes de trabalho, respectivamente, para cada conjunto de instâncias. De acordo com Aquino et al. (2018a), apenas a instância com 33.484 manutenções é instância do mundo real, que está incluída no conjunto $\mathcal{GG}$. Essa instância refere-se aos dados de planejamento de 52 semanas. Os autores geraram as outras instâncias decompondo a instância do mundo real. Eles definiram o valor de penalidade w_i para cada manutenção i com base na equação: $w_i = p_i \times prioridade_i$, em que p_i é o tempo de processamento da manutenção i e $prioridade_i$ é sua prioridade. Eles consideraram quatro níveis de prioridade ($\{40, 30, 20, 10\}$), sendo 40 associado à manutenção de maior prioridade e 10 à manutenção de menor prioridade.

Tabela 3.4: Características das instâncias.

conj.	#instâncias	#manutenções	#máquinas	#equipes
$\mathcal{P}$	100	20-80	2-5	2-14
$\mathcal{M}$	18	150-600	57-256	91-388
$\mathcal{G}$	18	1200-4800	88-283	252-1286
$\mathcal{GG}$	3	9600-33484	132-145	816-1032

3.8.2 Comparação dos métodos exatos

Está seção apresenta os resultados computacionais dos métodos exatos. Ela é dividida em duas partes. A primeira é apresenta os resultados para as instâncias do conjunto $\mathcal{P}$ e a segundo é a respeito os resultados da instância $\mathcal{M}$, como definido na Tabela 3.4.

3.8.2.1 Resultados para as instâncias do conjunto $\mathcal{P}$

Esta seção foca em comparar os resultados obtidos pela formulação NET, proposta por Aquino et al. (2018a), e pelas formulações TI e TIP, propostas neste trabalho, para as instâncias do conjunto $\mathcal{P}$. Em todos os experimentos desta seção, o tempo limite de execução do solver CPLEX foi fixado em 3600 segundos.

As Tabelas 3.5, 3.6 e 3.7 apresentam os resultados computacionais das formulações NET, TI e TIP, respectivamente, para os conjuntos de instâncias do conjunto $\mathcal{P}$. Nessas tabelas, as instâncias são classificadas segundo o número de máquinas. Assim, são agrupadas em quatro classes, com 25 instâncias em cada classe. Nas tabelas citadas, $|M|$ indica o número de máquinas considerados nas instâncias. Para as instâncias com mesmo número de máquinas,

$t(s)$ indica a média do tempo computacional de execução do *solver* em segundos; $gap(\%)$ indica a média do *gap* entre o melhor limitante superior (UB) e o melhor limitante inferior (LB) encontrado pelo *solver*, sendo dado por: $gap = 100(UB - LB)/UB$; z_{RL} indica a média dos limites inferiores dados pela relaxação linear; t_{RL} indica o tempo computacional de execução do *solver* em segundos para solucionar a formulação relaxada; e #opt indica o número de soluções ótimas encontradas. Como o tempo de execução das manutenções do conjunto de instâncias são fracionários e múltiplos de $0,1$, cada período da formulação TI tem a duração de $0,1$ horas. Como demonstrado na Seção 3.6, o número de períodos, que depende do parâmetro π, interfere no número de restrições e de variáveis de decisão. Assim, para analisar a influência de π no desempenho da formulação TIP para solucionar o conjunto P de instâncias, foram realizados experimentos computacionais para cinco valores de π. Os testes computacionais utilizaram os seguintes valores para o parâmetro π: 4, 8, 12, 18, e 24. Para facilitar a apresentação, a formulação TIP associada a um certo valor de π será denotada por FTIP$_\pi$. A Tabela 3.7 apresenta uma coluna extra para especificar o valor do parâmetro π. A Tabela 3.8, por sua vez, resume os resultados computacionais das formulações em todas as instâncias do conjunto $\mathcal{P}$.

Tabela 3.5: Resultados dos experimentos computacionais utilizando a formulação NET para as instâncias do conjunto $\mathcal{P}$.

	FN				
$\|M\|$	$t(s)$	$gap(\%)$	z_{RL}	$t_{RL}(s)$	#opt
2	1615.71	26.05	0.00	0.11	14
3	2076.16	25.27	0.20	0.13	12
4	3200.58	35.68	0.20	0.09	3
5	3170.82	24.53	12.68	0.06	4

Tabela 3.6: Resultados dos experimentos computacionais utilizando a formulação TI para as instâncias do conjunto $\mathcal{P}$.

	FTI				
$\|M\|$	$t(s)$	$gap(\%)$	z_{RL}	$t_{RL}(s)$	#opt
2	670.93	0.09	278.48	2.42	21
3	1397.94	3.14	826.68	3.09	16
4	1603.29	11.71	466.68	4.02	16
5	1611.46	3.92	510.88	6.18	16

Os resultados apresentados nas Tabelas 3.5, 3.6, 3.7 e 3.8 mostram um desempenho inferior da formulação NET em relação às formulações TI e TIP. Este resultado pode ser confirmado pelas médias de tempo computacional, que estão próximas do tempo limite de execução, pelas médias dos *gaps*, que são superiores a 24%, e o baixo número de soluções ótimas encontrada pela formulação proposta por Aquino et al. (2018a).

Tabela 3.7: Resultados dos experimentos computacionais utilizando a formulação TIP para as instâncias do conjunto $\mathcal{P}$.

		FTIP$_\pi$				
	$\|M\|$	$t(s)$	$gap(\%)$	z_{RL}	$t_{RL}(s)$	#opt
$\pi = 4$	2	1416.54	22.06	201.68	9.20	18
	3	1724.96	19.91	746.36	7.39	16
	4	1793.84	25.83	398.52	5.75	15
	5	1603.26	22.27	441.12	4.66	15
$\pi = 8$	2	316.40	1.33	201.68	3.40	24
	3	328.95	0.14	734.52	2.58	24
	4	454.60	2.55	398.52	2.22	23
	5	451.01	1.87	441.12	1.65	23
$\pi = 12$	2	368.74	1.32	192.08	2.26	24
	3	550.25	0.08	718.20	1.74	23
	4	761.38	4.13	382.20	1.53	21
	5	615.93	2.44	414.88	1.10	22
$\pi = 18$	2	158.87	0.00	74.00	1.29	25
	3	675.53	0.90	567.00	1.06	23
	4	596.20	1.61	256.92	0.92	21
	5	491.47	0.60	297.28	0.66	23
$\pi = 24$	2	171.61	1.00	192.08	0.90	24
	3	277.55	0.12	638.52	0.77	24
	4	270.09	0.10	363.96	0.60	24
	5	189.73	0.26	403.36	0.47	24

Tabela 3.8: Resumo dos resultados computacionais das formulações para as instâncias do conjunto $\mathcal{P}$.

	$t(s)$	$gap(\%)$	z_{RL}	$t_{RL}(s)$	#opt
FN	2515.82	27.88	3.27	**0.10**	33
FTI	1320.91	4.72	**520.68**	3.93	69
FTIP$_4$	1634.65	22.52	446.92	6.75	64
FTIP$_8$	387.74	1.47	443.96	2.46	94
FTIP$_{12}$	574.08	1.99	426.84	1.66	90
FTIP$_{18}$	480.52	0.78	298.80	0.98	92
FTIP$_{24}$	**227.25**	**0.37**	399.48	0.69	**96**

A Tabela 3.5 mostra que a maior parte das soluções ótimas encontradas pela formulação NET são instâncias com $|M|$ igual a 2 e 3 e a menor parte são instâncias com $|M|$ igual a 4 e 5. Como pode ser observado na Tabela 3.8, a formulação NET encontrou soluções ótimas em

apenas 33% das instâncias do conjunto $\mathcal{P}$, enquanto que a formulação TI encontrou soluções ótimas em 69% das instâncias. Os resultados demonstram que a formulação baseada em variáveis indexada no tempo, proposta neste trabalho, obteve melhores resultados que a formulação proposta em Aquino et al. (2018a). Além disso, os valores médios da coluna *gaps* da Tabela 3.6 mostram uma diferença na qualidade das soluções encontradas pelas formulações. Os bons valores de *gaps* da formulação TI possivelmente estão vinculados aos bons limites inferiores gerados pela relaxação linear da formulação, como demonstra os valores da coluna z_{RL}.

Os resultados reportados nas tabelas indicam que os melhores resultados estão associados à formulação FTIP_π proposta nesta tese. A FTIP_{24} foi capaz de encontrar solução ótima para 96% das instâncias do conjunto $\mathcal{P}$. Para outros valores de π, a porcentagem de soluções ótimas encontradas varia, porém permanece superior a 90%, com exceção para a formulação com $\pi = 4$. Os experimentos usando a formulação FTIP_π demonstram que o parâmetro π gera resultados diferentes. Além do número de soluções ótimas encontradas, observa-se diferenças nas coluna $t_{RL}(s)$, no qual as médias são maiores nas formulações com menor valor de π. Uma possível justificativa para isso está associada à dimensão da formulação, uma vez que menores valores de π geram formulações com número maior de restrições e variáveis, que, consequentemente, necessitam de um tempo maior para serem solucionadas pelo *solver*. Outra diferença que pode ser destacada é o valor dos limites inferiores dados pela relaxação linear. Nota-se que as formulações com $\pi = 4$ e $\pi = 8$ obtiveram limites inferiores melhores, enquanto que a formulação com $\pi = 18$ obteve limites inferiores piores. Embora a formulação com maior valor de π obteve os melhores resultados, a FTIP_8 encontrou o segundo maior número de soluções ótimas, em vez da FTIP_{12} e da FTIP_{18}. Além disso, a FTIP_4 gera bons limites inferiores; no entanto, é uma formulação de dimensão maior que as demais, e encontrou um número menor de soluções ótimas que a formulação TI. Vale destacar que, com a combinação dos melhores resultados da formulação TIP para todos os valores de π, foi possível encontrar solução ótima para todas as instâncias do conjunto $\mathcal{P}$.

A comparação dos valores na coluna z_{RL} da Tabela 3.8 mostra que a formulação FTI gera os melhores limites inferiores, enquanto que a formulação NET obtém os piores. Esses dois resultados estão de acordo com a literatura, quando se considera os dois tipos de formulação aplicadas em outros problemas de sequenciamento de máquinas. Com relação à formulação TIP, ela gera bons limites inferiores com relativa proximidade com a formulação TI. Para todas as formulações geradas com diferentes valores de π, nota-se que as médias dos limites inferiores são melhores que as médias dos limites inferiores referentes à formulação NET.

Com relação à comparação dos tempos computacionais para solucionar as formulações relaxadas, os valores na coluna t_{RL} da Tabela 3.8 mostram que as menores médias de tempo estão associadas à formulação NET. Como esperado, a formulação TI possui médias de tempo computacional com valores superiores ao da formulação NET. Os tempos computacionais da formulação TIP variam de acordo com os valores de π, em que as maiores médias de tempo estão associadas às formulações com menores valores de π. A média de tempo da

$FTIP_4$ chega a ser superior ao da formulação TI. No entanto, a relaxação linear das $FTIP_{24}$ e $FTIP_{18}$ foi resolvida em tempos bem inferiores ao tempo de resolução da relaxação linear da formulação TI e relativamente próximo ao da formulação NET.

A Figura 3.6 foi elaborada com o objetivo de apresentar uma visão alternativa dos *gaps* obtidos pelas formulações. Esta figura apresenta a porcentagem de ocorrência de *gaps* em diferentes intervalos para cada formulação. Nesta figura, nota-se que a formulação TI encontrou, na maior parte das instâncias, soluções com *gap* inferior a 1%. No entanto, para 27% das instâncias, o *gap* foi superior a 50%. Além disso, verifica-se que as instâncias em que a $FTIP_\pi$ com $\pi \geq 8$ não encontra solução ótima, obtém-se um *gap* em sua maioria superior a 1%. Este resultado pode estar relacionado com a capacidade da formulação TIP gerar bons limites superiores e a dificuldade em gerar limites inferiores que provem a otimalidade da solução.

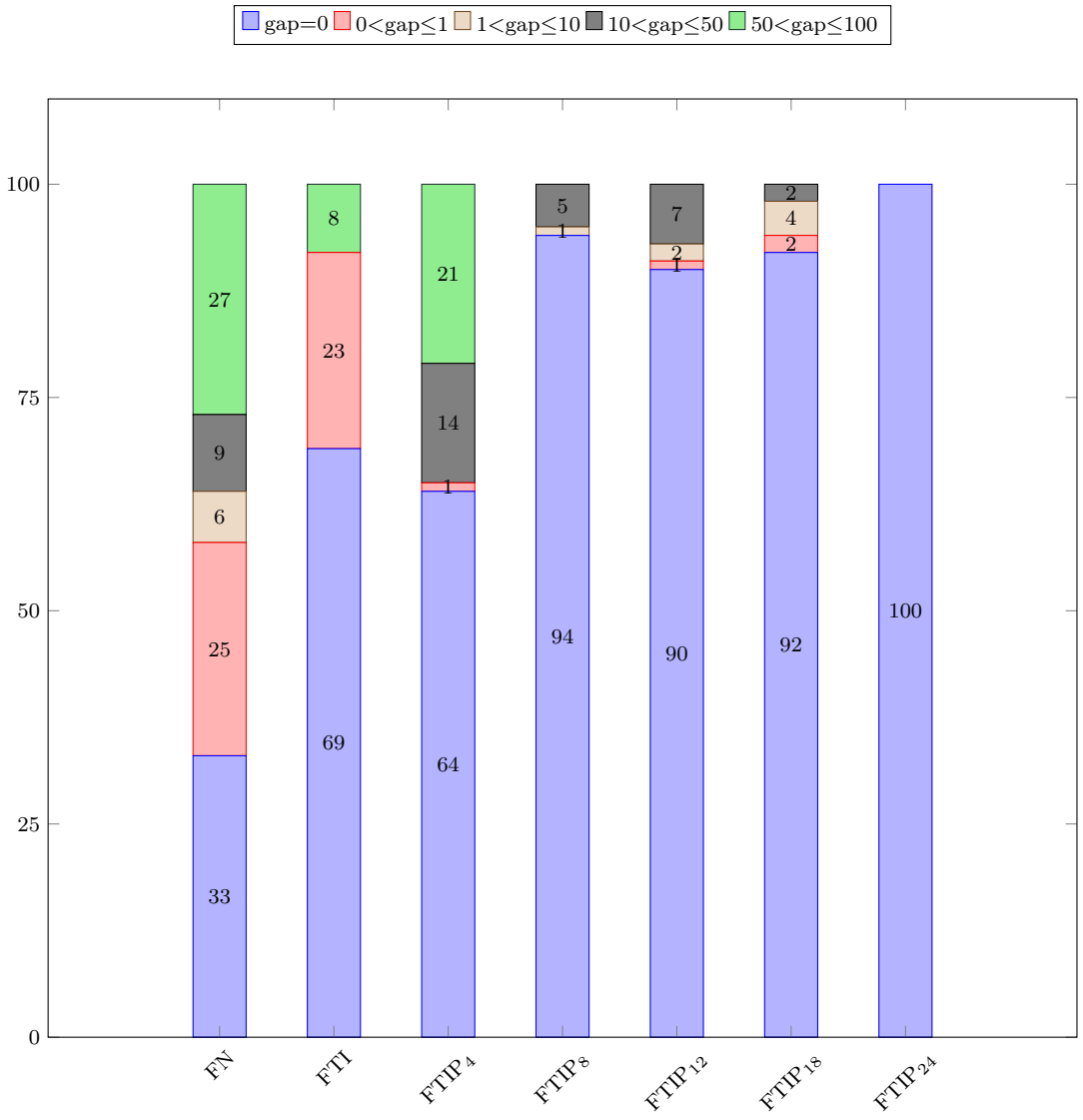


Figura 3.6: Gráfico de barra com *gaps* gerados pelas formulações para as instâncias do conjunto $\mathcal{P}$.

Além dos testes apenas com as formulações, testes adicionais foram conduzidos com um método exato híbrido, combinando cada formulação com o algoritmo GRASP. Este método de resolução é referenciado nesta tese como GRASP+Formulação. Como a solução GRASP fornece o limite superior inicial, a heurística do solucionador CPLEX foi desabilitada.

Para as instâncias do conjunto $\mathcal{P}$, a busca local do algoritmo GRASP utiliza a estratégia *best improvement*, uma vez que as instâncias são de pequena escala. Portanto, apenas os parâmetros de entrada do GRASP devem ser calibrados. Para isso, utilizamos a ferramenta IRACE (López–Ibáñez et al., 2016). Esta ferramenta foi projetada para encontrar o conjunto de valores mais adequado para os parâmetros de um algoritmo a partir de um conjunto de valores previamente fornecido a ela, juntamente com um conjunto de instâncias representativas do problema. A Tabela 3.9 mostra os valores fornecidos à ferramenta IRACE e o valor retornado para cada parâmetro do algoritmo aplicado às instâncias do conjunto $\mathcal{P}$.

Tabela 3.9: Parâmetros, conjunto de valores fornecidos ao IRACE e valores retornados por ele para instâncias do conjunto $\mathcal{P}$.

Parâmetros	**Conjunto de valores fornecidos ao IRACE**	**Valores retornados**
GRASPmax	$\{25, 40, 50\}$	50
α	$\{\lfloor 0.15 \times \lvert I\rvert \rfloor, \lfloor 0.25 \times \lvert I\rvert \rfloor, \lfloor 0.5 \times \lvert I\rvert \rfloor\}$	$\lfloor 0.25 \times \lvert I\rvert \rfloor$

A Tabela 3.10 apresenta os resultados das versões, combinando cada formulação com o algoritmo GRASP para as instâncias do conjunto $\mathcal{P}$. Cada versão GRASP+Formulação foi executada cinco vezes para cada instância. Além da coluna $\#opt$, esta tabela mostra $t_{best}(s)$, que indica o tempo computacional médio para executar cada versão GRASP+Formulação, e $gap_{best}(\%)$, que indica a diferença média entre o melhor limite superior e o melhor limite inferior encontrados pelo solucionador CPLEX a partir da melhor solução. Os valores médios dessas duas colunas foram calculados para cada formulação, considerando as 100 instâncias do conjunto $\mathcal{S}$ e as cinco execuções, e apenas o melhor resultado para cada instância é reportado.

Tabela 3.10: Resultados computacionais de cada método GRASP+Formulação para as instâncias do conjunto $\mathcal{P}$.

	GRASP+Formulação		
	$t_{best}(s)$	$gap_{best}(\%)$	#opt
FN	2589.67	26.65	33
FTI	1270.91	1.08	71
FTIP$_4$	1594.36	9.10	65
FTIP$_8$	383.12	1.75	93
FTIP$_{12}$	671.83	2.68	89
FTIP$_{18}$	608.94	1.64	93
FTIP$_{24}$	212.61	0.37	96

A comparação das Tabelas 3.8 e 3.10 mostra melhorias nos resultados nas formulações

TI, NET, FTIP_4, FTIP_{18} e FTIP_{24} com a implementação do método GRASP+Formulação. Na formulação TI, houve um aumento no número de soluções ótimas e na média do *gap*. A formulação NET também obteve melhora na média do *gap*; no entanto, o número de soluções ótimas encontradas continua o mesmo. A melhoria nos *gaps* pode ser resultado dos bons limites superiores iniciais obtidos com a heurística GRASP. Os bons limites superiores da heurística GRASP combinado com os bons limites inferiores gerados pela formulação TI, resultaram em melhoria nos valores de *gap*. No caso da formulação NET, a média do *gap* continua superior a 20%, devido aos fracos limites inferiores gerados pela formulação. Com relação à formulação TIP, nota-se uma melhora no número de soluções ótimas encontradas nas formulações com $\pi = 4$ e $\pi = 18$, enquanto para $\pi = 12$ obteve-se uma redução no número de soluções ótimas e para $\pi = 24$ manteve-se o número.

3.8.2.2 Resultados para as instâncias do conjunto $\mathcal{M}$

Esta seção foca em comparar os resultados obtidos pelas formulações NET, TI e TIP nas instâncias do conjunto $\mathcal{M}$. Em todos os experimentos desta seção, o tempo limite de execução do *solver CPLEX* foi fixado em 3600 segundos.

A Tabela 3.12 apresenta os resultados computacionais para as instâncias do conjunto $\mathcal{M}$ para dois métodos de solução. Um método utiliza apenas as formulações apresentadas para solucionar o problema, enquanto o segundo usa o método GRASP+Formulação. Além das colunas $t(s)$, $gap(\%)$, z_{RL}, $\#opt$, $t_{best}(s)$ e $gap_{best}(\%)$, a Tabela 3.12 apresenta a coluna $\#oom_1$, que indica o número de instâncias para as quais o *solver* não foi capaz de obter uma solução para o problema por falta de memória, e a coluna $\#oom_2$, que indica o número de instâncias para as quais a execução do *solver* foi finalizada antes de atingir o tempo limite por falta de memória. Como as melhores soluções encontradas pela $FTIP_\pi$ na resolução das instâncias do conjunto $\mathcal{P}$ foram obtidas para $\pi = 24$, que foi o maior valor de π testado, então nestes experimentos foram utilizados os seguintes valores de π: 24, 48 e 72. Para cada instância, o método GRASP+Formulação foi executada 5 vezes. Para o conjunto $\mathcal{M}$, a busca local da GRASP utiliza o método da descida randômica.

Portanto, a ferramenta IRACE deve calibrar o parâmetro *RDmax*, além dos parâmetros do GRASP. Seis instâncias do conjunto $\mathcal{M}$ foram usadas para calibrar seus valores de parâmetros. A Tabela 3.11 mostra os valores fornecidos à ferramenta IRACE e o valor retornado para cada parâmetro do algoritmo GRASP para as instâncias do conjunto $\mathcal{M}$.

Tabela 3.11: Resultados computacionais de cada método GRASP+Formulação para as instâncias do conjunto $\mathcal{M}$.

Parâmetros	**Conjunto de valores fornecidos ao IRACE**	**Valores retornados**
GRASPmax	$\{25, 40, 50\}$	40
α	$\{\lfloor 0.15 \times \lvert I\rvert \rfloor, \lfloor 0.25 \times \lvert I\rvert \rfloor, \lfloor 0.5 \times \lvert I\rvert \rfloor\}$	$\lfloor 0.5 \times \lvert I\rvert \rfloor$
RDmax	$\{250, 400, 500\}$	250

Os resultados mostram que a formulação TIP foi capaz de encontrar soluções ótimas, enquanto que as demais formulações não encontraram nenhuma solução ótima. Os valores

Tabela 3.12: Resultados computacionais para as instâncias do conjunto $\mathcal{M}$.

	Formulação				**GRASP+Formulação**			$\#oom_1$	$\#oom_2$
	$t(s)$	$gap(\%)$	z_{RL}	#opt	$t_{best}(s)$	$gap_{best}(\%)$	#opt		
FN	3601.18	42.61	859.38	0	3600.90	25.42	0	8	5
FTI	–	–	–	0	–	–	0	18	0
FTIP$_{24}$	1971.25	20.53	1131.50	7	1455.43	6.15	7	6	0
FTIP$_{48}$	2342.96	24.41	1351.71	5	2214.92	7.42	5	4	1
FTIP$_{72}$	2224.26	29.28	1804.25	6	1852.31	1.95	7	2	1

não nulos na coluna $\#oom_1$ e o símbolo (-) nas demais colunas associada à formulação TI nos dois métodos indicam que o *solver* não conseguiu gerar nenhuma solução para o problema. Esse resultado é devido à escala das instâncias, que gera formulações com dimensões maiores e grande número de períodos, que requerem uma quantidade superior de memória.

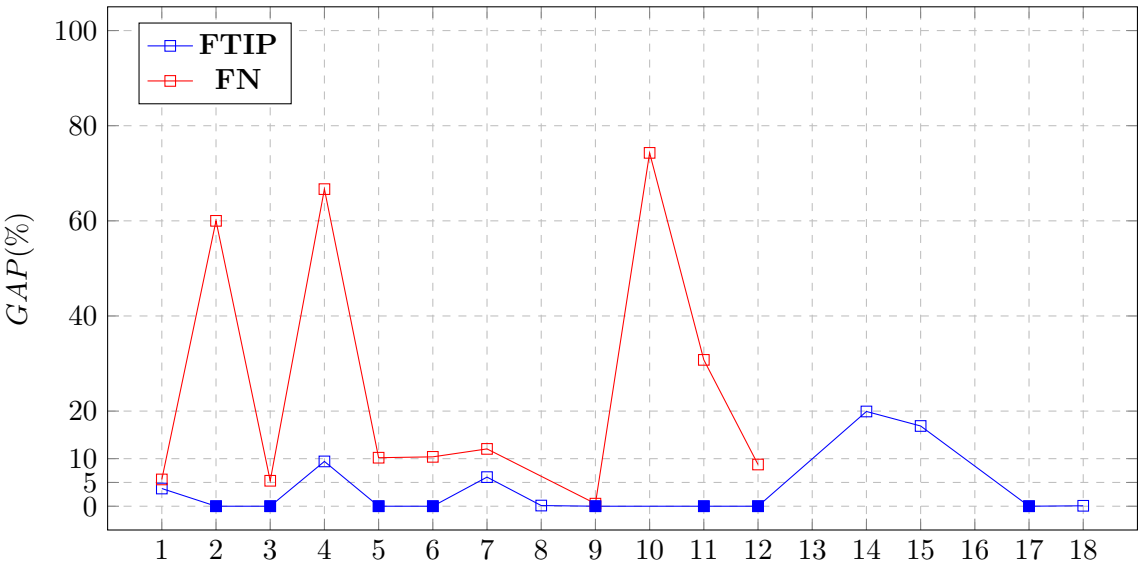
Com relação à formulação NET, ela não encontrou nenhuma solução ótima, porém gerou resultados até o tempo limite considerado para $27,78\%$ das instâncias. O método híbrido exato gerou *gaps* melhores, porém nenhuma solução ótima foi encontrada. Na coluna $\#oom_2$, verifica-se que em 5 instâncias o *solver* finalizou antes de atingir o tempo limite. Isso acontece porque a formulação NET gera um grande número de nós, que requer uma grande capacidade de memória. Ao atingir a capacidade limite de memória do computador, o *solver* é finalizado.

Em ambos os métodos, a FTIP$_{72}$ conseguiu gerar soluções para instâncias que as FTIP$_{24}$ e FTIP$_{48}$ não conseguiram. Além disso, verifica-se na tabela que a FTIP$_{72}$ apresenta o menor valor na coluna $\#om_1$. A justificativa para isso está associada ao parâmetro π. Com maiores valores de π, gera-se formulações com dimensões menores, que, consequentemente, requerem menor valores de memória. Quando são comparados os *gaps* dos dois métodos para a formulação TIP, nota-se uma melhora nos valores das médias. Essa melhoria pode estar associada à combinação dos bons limites inferiores gerados pela formulação TIP e os bons limites superiores dados pela heurística GRASP.

A Figura 3.7 apresenta o melhor *gap* para cada instância do conjunto M obtido pelas formulações NET, FTIP$_{24}$, FTIP$_{48}$ e FTIP$_{72}$. Como não foi possível gerar soluções para a formulação TI, ela não aparece na figura. O melhor *gap* é dado pelo menor valor obtido entre os métodos com e sem o uso da heurística GRASP. Além disso, no caso da formulação TIP, o melhor *gap* é o menor valor entre os dois métodos para FTIP$_{24}$, FTIP$_{48}$ e FTIP$_{72}$. Como mostra a Tabela 3.4, são consideradas 18 instâncias no conjunto M. Nesta figura, cada quadrado refere-se ao valor do *gap* para uma instância e o quadrado preenchido indica que a respectiva formulação encontrou a solução ótima para respectiva instância. Além disso, nota-se que algumas instâncias não possuem nenhum quadrado associado; nesses casos, a formulação não conseguiu gerar nenhuma solução por falta de memória.

Com a combinação dos resultados, a formulação TIP encontrou solução ótima em 8 instâncias, como mostra a Figura 3.7. Além disso, essa formulação obteve um *gap* menor que

Figura 3.7: Figura com os melhores gaps obtidos pelos métodos das formulações NET e TIP para as instâncias do conjunto $\mathcal{M}$.



10% em 13 instâncias. Na instância 9, os quadrados das duas formulações estão sobrepostos; no entanto, a formulação TIP encontrou a solução ótima e a formulação NET obteve um *gap* de 0.54%. É importante destacar que apenas a FTIP_{72} foi capaz de gerar soluções para instâncias com 600 manutenções. Esta formulação foi capaz de encontrar solução ótima para uma instância e para outra obteve um *gap* de $0,09$.

3.8.3 Comparação entre as soluções heurísticas

Esta seção é dedicada a comparar os limites superiores das instâncias do conjunto $\mathcal{M}$, $\mathcal{G}$ e $\mathcal{GG}$, encontrados nos trabalhos de Aquino et al. (2018a,b, 2019) e Santos et al. (2022), com os providos pelo algoritmo GRASP proposto neste trabalho.

Na Tabela 3.15 são apresentados e comparados os resultados computacionais do algoritmo GRASP proposto com os melhores resultados encontrados em Aquino et al. (2019), Santos et al. (2022) e Coelho et al. (2026). Nesta tabela, as colunas *Conjunto* e *id* identificam o conjunto e a instância, respectivamente. As colunas $|I|$, $|K|$ e $|M|$ representam o número de manutenções, máquinas e equipes de trabalho, respectivamente, disponível na instância. A coluna BKS indicam o melhor limite superior conhecido para cada instâncias, considerando o valor encontrado na presente tese e nos demais estudos da literatura. As colunas z e $t(s)$ estão associadas às soluções dadas pelos algoritmos meta-heurísticos da literatura e representam o melhor limite superior encontrado e o tempo de execução em segundos do algoritmo, respectivamente. Como os experimentos computacionais de Aquino et al. (2018a,b, 2019), Santos et al. (2022) e Coelho et al. (2026) usam uma máquina diferente daquela usada neste trabalho, os tempos computacionais foram ajustados usando-se o *software* `PassmarMark`, disponível em `https://www.cpubenchmark.net/singleCompare.php`, permitindo a comparação entre os processadores de cada computador usado. Ao dividir o

valor de `CPU Maker` do computador utilizado no respectivo trabalho pelo valor do `CPU Maker` do computador utilizado nos experimentos computacionais desta tese, obtém-se um fator de ajuste. Para Aquino et al. (2018a,b, 2019), esse valor é 1,44; para Santos et al. (2022), esse valor é dado por é 0,98; e para Coelho et al. (2026) esse valor é 2,90. Portanto, os tempos da coluna $t(s)$ são iguais aos tempos de execução apresentados nos respectivos trabalhos multiplicados pelo fator de ajuste. As colunas $z_{\min}$ e $z_{média}$ apresentam o limite superior gerado pelo algoritmo GRASP e representam o valor mínimo e valor médio, respectivamente. O valor mínimo encontrado pelo algoritmo GRASP representa a melhor solução. A coluna $t_{média}$ representa o tempo médio, em segundos, em cinco execuções, do algoritmo em cada instância. A coluna RPD(%) mostra o *Relative Percentage Deviation*(RPD). Esse indicador compara o limite superior da coluna BKS com o melhor limite superior encontrado no trabalho associado a coluna. O valor de RPD é dado por:

$$RPD = 100 \times \frac{z - z_{BKS}}{z_{BKS}}$$

Assim, $RPD > 0$ indica que o limite superior encontrado pelo algoritmo do trabalho é maior do que o melhor conhecido na respectiva instância. Se $RPD = 0$, indica que o limite superior gerado pelo algoritmo do trabalho é igual ao melhor limite superior conhecido. O carácter --"indica que o trabalho citado não apresentou uma solução para a instância. Além disso,

Os valores dos parâmetros utilizados pelo algoritmo para as instâncias do conjunto $\mathcal{M}$ são os mesmos da Subseção 3.8.2.2. Seis instâncias foram utilizadas para calibrar os valores dos parâmetros do algoritmo GRASP para o conjunto $\mathcal{G}$ e três instâncias para o conjunto $\mathcal{GG}$. As tabelas 3.13 e 3.14 mostram o conjunto de valores fornecidos à ferramenta IRACE e o valor retornado para cada parâmetro do algoritmo em instâncias dos conjuntos $\mathcal{L}$ e $\mathcal{GG}$, respectivamente.

Tabela 3.13: Conjunto de valores fornecidos ao IRACE e valores retornados por ele para cada parâmetro do GRASP aplicado ao conjunto $\mathcal{G}$.

Parâmetros	**Conjunto de valores fornecidos ao IRACE**	**Valores retornados**
$GRASPmax$	$\{25, 40, 50\}$	40
α	$\{\lfloor 0.15 \times \lvert I\rvert \rfloor, \lfloor 0.25 \times \lvert I\rvert \rfloor, \lfloor 0.5 \times \lvert I\rvert \rfloor\}$	$\lfloor 0.25 \times \lvert I\rvert \rfloor$
$RDmax$	$\{250, 400, 500\}$	400

Tabela 3.14: Conjunto de valores fornecidos ao IRACE e valores retornados por ele para cada parâmetro do GRASP aplicado ao conjunto $\mathcal{GG}$.

Parâmetros	**Conjunto de valores fornecidos ao IRACE**	**Valores retornados**
$GRASPmax$	$\{25, 40, 50\}$	40
α	$\{\lfloor 0.1 \times \lvert I\rvert \rfloor, \lfloor 0.15 \times \lvert I\rvert \rfloor, \lfloor 0.25 \times \lvert I\rvert \rfloor\}$	$\lfloor 0.1 \times \lvert I\rvert \rfloor$
$RDmax$	$\{400, 500, 750\}$	500

De acordo com Santos et al. (2022), as soluções encontradas por Aquino et al. (2019) nas

instâncias 1, 7, 13, 19, 25 e 31 são inviáveis; logo, seus resultados não serão considerados. Por causa disso, estas instâncias possuem o símbolo $--*$ nas colunas z_{bks}, t_{BKS} e $gap_1(\%)$.

Tabela 3.15: Resultados providos pelo algoritmo GRASP nas instâncias dos conjuntos M, G e GG. As

						GRASP*				Aquino et al. (2018a,b, 2019)**			Santos et al. (2022)***			Coelho et al. (2026)***		
Conjunto	**id**	$\|I\|$	$\|K\|$	$\|E\|$	z_{BKS}	z	z_{mdia}	t_{mdia}	**RPD**	z	$t(s)$	**RPD**	z	$t(s)$	**RPD**	z	$t(s)$	**RPD**
$\mathcal{M}$	1	150	120	148	1848	**1848**	1848	0,09	**0,00**	–*	–*	–*	1850	147,9	0,11	**1848**	435,37	**0,00**
	2	150	129	75	**30**	**30**	30	0,06	**0,00**	30	216	**0,00**	30	147,9	**0,00**	30	435,37	**0,00**
	3	150	91	102	**3273**	**3273**	3273	0,07	**0,00**	3273	216	**0,00**	3273	147,9	**0,00**	3273	435,37	**0,00**
	4	150	126	57	**24**	**24**	24	0,12	**0,00**	24	216	**0,00**	24	147,9	**0,00**	24	435,37	**0,00**
	5	150	93	92	**49**	**49**	49	0,05	**0,00**	49	216	**0,00**	49	147,9	**0,00**	49	435,37	**0,00**
	6	150	101	71	**106**	**106**	106	0,06	**0,00**	106	216	**0,00**	106	147,9	**0,00**	106	435,37	**0,00**
	7	300	179	158	**1931**	**1931**	1985,4	0,18	**0,00**	–*	–*	–*	1991	295,8	3,11	1931	870,73	**0,00**
	8	300	239	221	**2451**	**2451**	2452,6	0,22	**0,00**	2452	432	0,04	2456	295,8	0,20	2451	870,73	**0,00**
	9	300	177	112	**3359**	**3359**	3359	0,17	**0,00**	3360	432	0,03	3363	295,8	0,12	3359	870,73	**0,00**
	10	300	181	75	**35**	**35**	35	0,23	**0,00**	35	432	**0,00**	35	295,8	**0,00**	35	870,73	**0,00**
	11	300	162	121	**65**	**65**	65	0,12	**0,00**	281	432	332,31	65	295,8	**0,00**	65	870,73	**0,00**
	12	300	176	119	**308**	**308**	308	0,14	**0,00**	308	432	**0,00**	308	295,8	**0,00**	308	870,73	**0,00**
	13	600	329	165	**4494**	4566	4624,2	0,96	1,60	–*	–*	–*	4684	591,6	4,23	**4494**	1741,47	**0,00**
	14	600	388	256	**3380**	3448	3457,6	0,83	2,01	3384	864	0,12	3465	591,6	2,51	**3380**	1741,47	**0,00**
	15	600	288	120	**8576**	**8576**	8974,4	1,20	**0,00**	8578	864	0,02	8580	591,6	0,05	8577	1741,47	0,01
	16	600	215	77	**37**	38	38	0,71	2,70	42	864	13,51	40	591,6	8,11	**37**	1741,47	**0,00**
	17	600	279	126	**410**	410	410	0,77	**0,00**	414	864	0,98	410	591,6	**0,00**	410	1741,47	**0,00**
	18	600	292	120	**5660**	**5660**	5660,6	0,50	**0,00**	5663	864	0,05	5959	591,6	5,28	–	–	–
$\mathcal{G}$	19	1200	186	519	**10921**	10923	10973,2	10,31	0,02	–*	–*	–*	11056	1183,2	1,24	**10921**	3482,93	**0,00**
	20	1200	263	666	**9517**	9621	9653,6	4,96	1,09	9527	1728	0,11	9657	1183,2	1,47	**9517**	3482,93	**0,00**
	21	1200	122	470	**21921**	22673	23068,2	5,69	3,43	21930	1728	0,04	23621	1183,2	7,76	**21921**	3482,93	**0,00**
	22	1200	88	252	**299**	**299**	299	2,87	**0,00**	309	1728	3,34	300	1183,2	0,33	**299**	3482,93	**0,00**
	23	1200	130	420	**513**	581	597,2	2,67	13,26	526	1728	2,53	529	1183,2	3,12	**513**	3482,93	**0,00**
	24	1200	122	403	**6829**	6830	6830,2	2,50	0,01	6841	1728	0,18	**6829**	1183,2	**0,00**	6836	3482,93	0,10
	25	2400	188	738	**26174**	**26174**	26314,8	61,52	**0,00**	–*	–*	–*	26665	2366,4	1,88	26932	6965,86	2,90
	26	2400	283	869	**24528**	**24528**	24631,4	38,42	**0,00**	24553	3456	0,10	24826	2366,4	1,21	–	–	–
	27	2400	130	603	**37566**	45537	45846,4	26,15	21,22	38094	3456	1,41	40773	2366,4	8,54	**37566**	6965,86	**0,00**
	28	2400	90	278	**566**	**566**	569,2	14,92	**0,00**	1585	3456	180,04	567	2366,4	0,18	570	6965,86	0,71
	29	2400	132	701	**621**	760	776,4	14,46	22,38	816	3456	31,40	715	2366,4	15,14	**621**	6965,86	**0,00**
	30	2400	126	561	**8234**	**8234**	8234,2	9,79	**0,00**	8259	3456	0,30	8474	2366,4	2,91	8242	6965,86	0,10
	31	4800	197	1128	**56164**	**56164**	56252,6	299,18	**0,00**	–*	–*	–*-	56564	4732,8	0,71	56902	13931,72	1,31
	32	4800	78	1286	**40953**	41135	41797,2	122,66	0,44	41925	6912	2,37	42735	4732,8	4,35	**40953**	13931,72	**0,00**
	33	4800	130	720	**86147**	91174	95693	177,82	5,84	92320	6912	7,17	93995	4732,8	9,11	**86147**	13931,72	**0,00**
	34	4800	91	294	**194868**	**194868**	194868,4	47,58	**0,00**	198340	6912	1,78	–	–	–	194944	13931,72	0,04
	35	4800	41	294	**1422**	1426	1426,6	54,80	0,28	–	–	–	**1422**	4732,8	**0,00**	–	–	–
	36	4800	135	990	**1944**	2613	2753,6	89,45	34,41	6008	6912	209,05	2130	4732,8	9,57	**1944**	13931,72	**0,00**
	37	4800	129	713	**15171**	16212	16737,6	60,23	6,86	16745	6912	10,38	16617	4732,8	9,53	**15171**	13931,72	**0,00**
$\mathcal{GG}$	38	9600	132	816	**29943**	33782	35119,2	349,19	12,82	35778	13824	19,49	30002	9465,6	0,20	**29943**	27863,45	**0,00**
	39	19200	132	908	**68711**	80720	81725,2	1.690,19	17,48	102773	27648	49,57	**68711**	18931,2	**0,00**	69688	55726,89	1,42
	40	33484	145	1032	**123538**	141431	149869	5.193,56	14,48	220048	48216,96	78,12	128856	33015,22	4,30	**123538**	97185,38	**0,00**

* Executado em um Intel Core i3-10100T 3.00 GHz x 8 com 16 GB de memória RAM;
** Executado em um Intel Xeon CPU E5-2660v2 2.20 GHz x 20 com 384 GB de memória RAM;
*** Executado em um Intel Core CPU i7-4790 3.60GHz x 8 com 15 GB de memória RAM;
**** Executado em um Intel Xeon E5-2699 v4 @ 2.20GHz x 44 com 64 GB de memória RAM

As instâncias destacadas em vermelho indicam as soluções que foram provadas serem ótimas nos métodos exatos da Seção 3.8.2.2 e que o algoritmo GRASP foi capaz de encontrar. Pelos valores destacados em negrito na Tabela 3.15 nas colunas z dos dois trabalhos citados e $z_{\min}$, nota-se que o algoritmo GRASP encontrou as melhores soluções em 60% das instâncias. Vale destacar a diferença significativa de tempo computacional do algoritmo GRASP proposto com relação aos algoritmos da literatura. Ao comparar os resultados dessa pesquisa com os algorítimos de Aquino et al. (2018a,b, 2019) verifica-se que houve uma melhora expressiva nos limites gerados pelo algoritmo GRASP. Isso ocorre, principalmente nas instâncias de maior dimensão (instâncias 38 e 39) e na instância real (instância 40). Os algoritmos de Aquino et al. (2018a,b, 2019) encontram soluções iguais as melhores conhecidas em 22,50% das instâncias, enquanto, os algoritmos de Santos et al. (2022) encontram 35.00%. Ambos trabalhos apresentaram uma porcentagem inferior ao do presente trabalho. Vale destacar que, Santos et al. (2022) encontrou alguns limites superiores melhor que os demais trabalhos citados. Por fim, o algoritmo de Coelho et al. (2026) encontrou 77,50% das melhores soluções encontradas.

Os novos limites superiores encontrados na maior parte das instâncias indicam que as novas soluções possuem um percentual menor de equipes ativadas e maior percentual de manutenções programadas. Em algumas instâncias, a diferença entre os limites superiores encontrados anteriormente e os novos é relativamente menor que as demais, e isto significa que, nessas novas soluções, as tarefas foram alocadas de um modo diferente, que permite um menor número de equipes de manutenção ativadas.

3.9 Conclusões parciais

Esta seção abordou métodos de solução para o LTPMOSP. Para resolver este desafiador problema de otimização, foram propostas duas novas formulações matemáticas e um algoritmo heurístico. A primeira formulação, denominada TI, é baseada em variáveis indexadas no tempo, enquanto a segunda formulação, chamada TIP, usa variáveis indexadas no tempo e de precedência e divide o horizonte de planejamento a partir de um parâmetro de entrada π. Até onde sabemos, esta segunda formulação é nova na literatura. O algoritmo desenvolvido para tratar instâncias de maior dimensão do problema é baseado na metaheurística GRASP. Esse algoritmo baseado em GRASP utiliza um novo algoritmo de posicionamento de manutenções. Além disso, também, foi proposto um método exato híbrido, que combina o algoritmo GRASP com uma formulação de programação matemática. Neste caso, o algoritmo GRASP é usado apenas para gerar a solução inicial para o solucionador.

As formulações, o método exato híbrido e o algoritmo GRASP foram testados em instâncias variando de 20 a 33.434 manutenções. As duas formulações propostas foram comparadas com a formulação NET apresentada em Aquino et al. (2018a). Dentre essas três formulações, a formulação TIP foi a que obteve o melhor desempenho, encontrando soluções ótimas para todas as instâncias de 20 a 80 manutenções. Além disso, com o método exato híbrido

usando a formulação TIP, foi possível encontrar soluções ótimas em 44,45% das instâncias com 150 a 600 manutenções. Em 72,23% das instâncias deste conjunto, este método gerou soluções com *gaps* menores que 10%. Por outro lado, a formulação TI também apresentou resultados melhores que os da formulação da literatura nas instâncias de pequena dimensão. Com o método exato híbrido utilizando a formulação TI, foi possível encontrar as soluções ótimas em 71% das instâncias de 20 a 80 manutenções. No entanto, a formulação TI não conseguiu gerar nenhum resultado para as instâncias com 150 a 600 manutenções devido ao tamanho da instância. Os resultados computacionais mostraram que o formulação TIP conseguiu gerar bons limites inferiores e, ao mesmo tempo, exigir menor tempo computacional para resolver o relaxamento linear da formulação que a formulação TI. Além disso, esses tempos são próximos aos da formulação NET.

Os resultados computacionais mostraram que o algoritmo GRASP proposto foi capaz de obter limites superiores melhores ou iguais aos da literatura em 60% das instâncias com 150 a 33434 manutenções, em um tempo computacional significativamente inferior. Além disso, o algoritmo conseguiu encontrar a solução ótima em diversas instâncias. Os novos e bons limites superiores obtidos estão associados principalmente à estratégia para obter o vetor solução pelo algoritmo GRASP e com o algoritmo PM. Este algoritmo é parte fundamental do algoritmo meta-heurístico e sua estratégia de tentar alocar as manutenções primeiramente no final da janela origina novas e diferentes soluções para o problema. Os baixos tempos computacionais estão justamente relacionados com a estratégia adotada, de aplicar a fase de busca local apenas na melhor das soluções construídas.

Capítulo 4

Problema de planejamento de ordens de manutenção de longo prazo com multi-habilidade

Este capítulo é dedicado ao estudo do Problema de planejamento de ordens de manutenção de longo prazo com multi-habilidade (LTPMOSP-MS). A Seção 4.1 descreve formalmente o LTPMOSP-MS. A Seção 4.1.1 apresenta um exemplo para auxliar na melhor descrição do LTPMOSP-MS, envolvendo uma instância de pequena dimensão do problema e sua respectiva solução. A Seção 4.2 introduz a formulação matemática proposta nesta tese para o LTPMOSP-MS. Como ele é um problema $\mathcal{NP}$-difícil (Pinedo, 2022), pode ser difícil encontrar a solução ótima ou uma solução de boa qualidade para instâncias do problema, conforme a dimensão das instâncias aumenta. Para contornar isso, foram desenvolvidas versões de uma heurística de duas fases. A Seção 4.3 apresenta três versões da heurística proposta nesta tese. Para analisar e comparar o desempenho da formulação apresentada na Seção 4.2 com as diferentes versões da heurística duas fases para um conjunto de instâncias de diferentes escalas, é elaborado um extenso experimento computacional. Os resultados são apresentados, analisados e discutidos na Seção 4.4. Por fim, a Seção 4.5 apresenta conclusões parciais.

4.1 Descrição do problema LTPMOSP-MS

O Problema de Planejamento de Manutenções Preventivas com Multi-habilidades (*Long-Term Preventive Maintenance Order Scheduling Problem with Multi-Skill* – LTPMOSP-MS) é formalmente definido a seguir.

Considere um conjunto I de manutenções, um conjunto M de máquinas submetidas às manutenções preventivas, um conjunto K de equipes que executam essas manutenções e um conjunto S de habilidades que as manutenções requerem para serem executadas. Cada manutenção $i \in I$ requer um tempo de processamento ρ_i e deve ser programada

dentro de uma janela de tempo $[a_i, b_i]$. Cada manutenção pode ser programada em qualquer instante dentro dessa janela, desde que ela comece e finalize no interior de sua própria janela de tempo. Se uma manutenção $i \in I$ não pode ser programada dentro da janela, ela é classificada como não programada e penalizada com o valor w_i na função objetivo. Cada manutenção $i \in I$ deve ser programada para ser realizada em uma máquina $e_i = m \in M$. O subconjunto $\Phi_m \subseteq I$ denota as manutenções que devem ser realizadas na máquina $m \in M$. Além disso, cada manutenção $i \in I$ requer um subconjunto $\Psi_i \subseteq S$ de habilidades distintas para ser executada. Para cada habilidade $s \in \Psi_i$ da manutenção i há um subconjunto $\Gamma_i^s \subseteq K$ de equipes que possuem a habilidade s. Como cada manutenção i requer um total de λ_i habilidades, é necessário atribuir a ela uma equipe para cada habilidade. A manutenção i só pode ser atribuída às equipes do subconjunto $\Gamma_i = \Gamma_i^{s_1} \cup \Gamma_i^{s_2} \cup \cdots \cup \Gamma_i^{s_i}$, pois apenas elas possuem as habilidades necessárias para executá-la.

Cada equipe $k \in K$ possui apenas uma habilidade $sk_k = s \in S$ para executar as manutenções. Pode-se ter mais de uma equipe com a mesma habilidade. Além disso, uma equipe $k \in K$ está disponível para a execução das manutenções somente dentro das janelas de tempo $[0, \bar{b}_k^{(1)}]$ e $[\bar{a}_k, \bar{b}_k^{(2)}]$. O subconjunto $\Omega_k \subseteq I$ denota as manutenções que a equipe $k \in K$ tem a habilidade para executar.

As janelas das manutenções e das equipes podem ocorrer em instantes de tempo diferentes; no entanto, cada manutenção deve ser programada nas equipes em uma janela resultante (JR) que considera ambas as janelas. Assim, cada manutenção i terá uma ou duas JR para cada equipe $k \in \Gamma_i$. Devida à duração e posição dessas janelas, cada manutenção pode ter uma ou duas JRs, como mostra a Figura 4.1. O conjunto Q denota as JRs e o subconjunto Q_{ik} denota as janelas de cada manutenção i para cada equipe k. Assim, $Q_{ik} = \{1\}$ indica que a manutenção i possui apenas a primeira JR (Figura 4.1(a)); $Q_{ik} = \{2\}$ indica que a manutenção i possui apenas a segunda JR (Figura 4.1(b)); e $Q_{ik} = \{1, 2\}$ mostra que a manutenção i possui as duas JRs (Figura 4.1(c)). Se a manutenção possui apenas a primeira JR, ela é dada por:

$$\left[\max(a_i, 0);\ \min(b_i, \bar{b}_k^{(1)})\right] = [\alpha_{ik}^{(1)}; \beta_{ik}^{(1)}]\,.$$

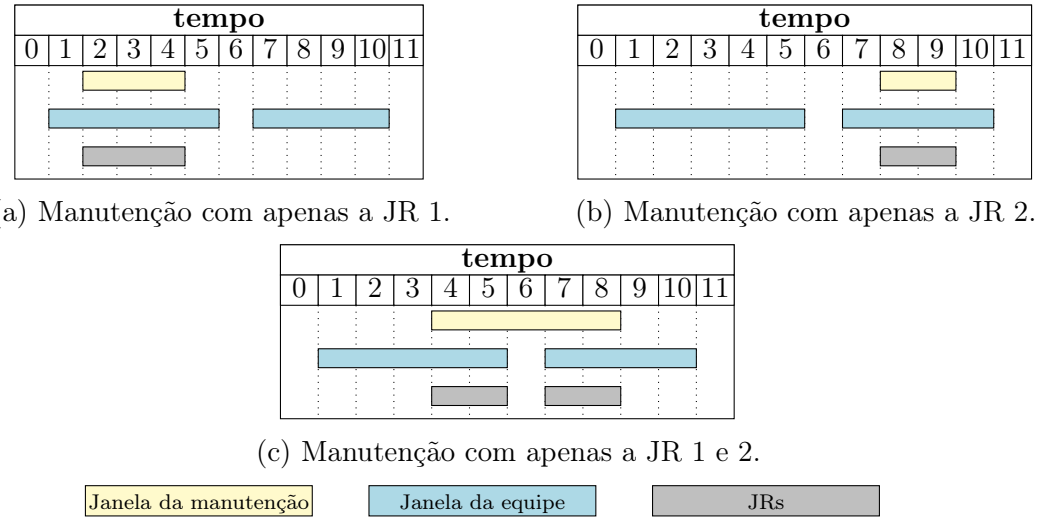
Se, por outro lado, a manutenção possui apenas a segunda JR, ela é dada por:

$$\left[\max(a_i, \bar{a}_k);\ \min(b_i, \bar{b}_k^{(2)})\right] = [\alpha_{ik}^{(2)}; \beta_{ik}^{(2)}]\,.$$

Caso a manutenção possua as duas JRs, elas são calculadas com as mesmas expressões e respectivamente.

O LTPMOSP-MS busca minimizar o número de equipes utilizadas no planejamento e a soma das penalidades das manutenções preventivas não programadas. As restrições do LTPMOSP-MS são as seguintes:

(i) Se programada, cada manutenção deve ser simultaneamente atribuída a uma máquina e às equipes. O instante de início de execução das manutenções preventivas deve ser o mesmo nas equipes e na máquina sob manutenção;



(a) Manutenção com apenas a JR 1.

(b) Manutenção com apenas a JR 2.

(c) Manutenção com apenas a JR 1 e 2.

Figura 4.1: Possibilidades de JRs para as manutenções.

(ii) Cada equipe pode executar apenas uma manutenção por vez;

(iii) Cada máquina sob manutenção não pode ter mais de uma manutenção sendo executada simultaneamente;

(iv) Cada manutenção i é executada por λ_i equipes que possuem as habilidades específicas para executá-la. A manutenção i deve ser atribuída a uma equipe $k \in \Gamma_i^s \subseteq K$ para cada habilidade $s \in \Psi_i \subseteq S$ que ela requer; e

(v) Cada manutenção $i \in I$ deve ser programada dentro de uma JR que considera a sua janela de execução e a janela da equipe $k \in K$.

4.1.1 Um exemplo numérico

Esta seção apresenta um exemplo do LTPMOSP-MS. O intuito é tornar mais claro o entendimento do problema. Para o exemplo, considere cinco manutenções preventivas $(1, 2, 3, 4, 5)$, três equipes $(1, 2, 3)$ e duas máquinas (A, B). Os dados de entrada para as manutenções preventivas e equipes utilizadas neste exemplo são mostrados nas Tabelas 4.1 e 4.2.

Tabela 4.1: Dados de entrada das manutenções.

i	e_i	ρ_i	$[a_i, b_i]$	w_i	Ψ_i
1	B	4	[3, 10]	160	{1,2}
2	B	3	[6, 12]	30	{2}
3	A	4	[15, 20]	90	{2,3}
4	B	4	[5, 15]	80	{3}
5	A	6	[0, 12]	120	{1,3}

A Figura 4.2 ilustra, utilizando um diagrama de Gantt, a atribuição das manutenções às equipes e máquinas. A barra na cor azul sobre os blocos para as manutenções representa as

Tabela 4.2: Dados de entrada das equipes.

k	sk_i	$[0, \bar{b}_i^{(1)}]$	$[\bar{a}_i, \bar{b}_i^{(2)}]$
1	1	[0, 10]	[11, 14]
2	2	[0, 3]	[5, 19]
3	3	[0, 14]	[15, 20]

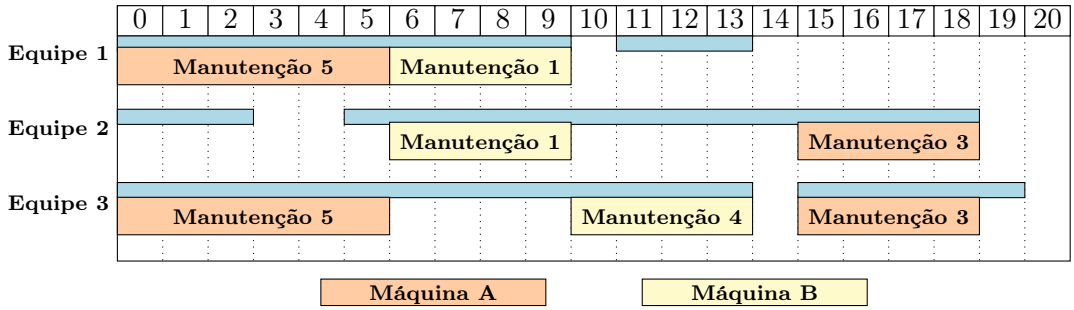


Figura 4.2: Ilustração de uma solução factível para o LTPMOSP-MS, representando a atribuição simultânea das manutenções a equipes e máquinas, assim como o tempo de início e fim das manutenções programadas.

janelas de tempo de cada equipe. Note que a Equipe 1 executa as manutenções 5 e 1, nesta sequência; a Equipe 2 executa as manutenções 1 e 3; e a Equipe 3 executa as manutenções 5, 4 e 3. Veja que cada uma das manutenções 1, 3 e 5 é programada para ser executada em um mesmo instante por duas equipes diferentes, porque elas requerem duas habilidades distintas para serem executadas.

Com relação ao sequenciamento nas máquinas, as manutenções 5 e 3 são programadas, nesta sequência, na Máquina A, enquanto as manutenções 1 e 4 são programadas na Máquina B. Note que as manutenções programadas na mesma máquina não são executadas ao mesmo tempo. Por fim, observa-se, na solução viável apresentada na Figura 4.2, que não é possível programar a manutenção 2 na Equipe 1. Isso ocorre porque as manutenções 1 e 4 estão programadas dentro da janela da manutenção 2 na Máquina B. Como o objetivo é minimizar as penalidades e a manutenção 2 possui o menor valor de penalidade dentre as três, ela não é programada. Através desse exemplo, nota-se que os sequenciamentos das manutenções nas equipes e máquinas estão integrados.

4.2 Formulação Matemática do LTPMOSP-MS

A formulação matemática proposta nesta tese para o LTPMOSP-MS é baseada em variáveis indexadas no tempo e é denotada por $\mathcal{TI}$. Nela, o horizonte de planejamento é dividido em períodos. A duração de cada período é ζ e é assumido que todos os parâmetros de entrada das manutenções e das equipes são múltiplos de ζ. O conjunto de períodos é definido como $P = \{0, \ldots, \mathcal{P}\}$, em que $\mathcal{P} = \max_{k \in K}\{\bar{b}_k^{(2)}\}/\zeta$. De forma geral, o que se pretende é definir em qual período a manutenção inicia sua execução, o que consequentemente

gera o sequenciamento das manutenções preventivas.

Para garantir que cada manutenção seja executada dentro da sua janela e das equipes, foram adicionadas limitações no índice $p \in P$ na formulação das restrições. Para isto, os períodos em que cada manutenção pode ser programada para iniciar sua execução para cada equipe são restritos na formulação para garantir que as restrições de janela de tempo sejam satisfeitas. Assim, considere $\bar{P}_i^k \subseteq P$ para denotar o subconjunto de períodos em que a manutenção i pode iniciar sua execução pela equipe k, para respeitar as janelas de tempo da manutenção e da equipe, o qual é dado por:

$$\bar{P}_{ik} = \Big\{ p \in P : p \in [\alpha_i/\zeta, \beta_i/\zeta - \rho_i/\zeta] \cap \big([0, b_k^{(1)}/\zeta - \rho_i/\zeta] \cup [a_k/\zeta, b_k^{(2)}/\zeta - \rho_i/\zeta]\big) \Big\}.$$

Além disso, é utilizado o parâmetro $\bar{s}_{ik}$ para representar o menor período de início de execução da manutenção i pela equipe k que pode estar em atividade no período p, em que:

$$\bar{s}_{ik}^p = \min\{\bar{p} \in \bar{P}_{ik} : \bar{p} \geq p - \rho_i/\zeta + 1\}.$$

A formulação $\mathcal{TI}$ utiliza os conjuntos, os parâmetros apresentados na Seção 4.1.1 e as seguintes variáveis binárias:

(i) y_k, que é igual a 1 se a equipe de trabalho k é ativada e 0, caso contrário, com $k \in K$;

(ii) v_i, que é igual a 1 se a manutenção i é executada e 0, caso contrário, com $i \in I$;

(iii) x_{ik}, que é igual a 1 se a manutenção $i \in I$ for programada para ser executada pela equipe de trabalho $k \in \Gamma_i$ e 0, caso contrário;

(iv) u_{ik}^p, que é igual a 1 se a manutenção $i \in I$ inicia sua execução com a equipe $k \in \Gamma_i$ no período $p \in \bar{P}_i^k$ e 0, caso contrário.

A formulação $\mathcal{TI}$ é dada, então, por:

$$\min \quad z = \sum_{k \in K} y_k + \sum_{i \in I} w_i (1 - v_i) \tag{4.1}$$

$$\textbf{suj. a:} \quad x_{ik} \leq y_k \qquad \forall i \in I, k \in \Gamma_i \tag{4.2}$$

$$\sum_{k \in \Gamma_i^s} x_{ik} = v_i \qquad \forall i \in I, s \in \Psi_i \tag{4.3}$$

$$\sum_{p \in \bar{P}_{ik}} u_{ik}^p \geq x_{ik} \qquad \forall i \in I, s \in \Psi_i, k \in \Gamma_i^s \tag{4.4}$$

$$\sum_{k \in \Gamma_i^s} \sum_{p \in \mathcal{P}_{ik}} u_{ik}^p = v_i \qquad \forall i \in I, s \in \Psi_i \tag{4.5}$$

$$\sum_{i \in \Omega_k} \sum_{\substack{l = s_{ik}^p \\ l \in \bar{P}_{ik}}}^{p} u_{ik}^l \leq 1 \qquad \forall k \in K, p \in P \tag{4.6}$$

$$\sum_{i\in\Phi_m}\sum_{k\in\Gamma_i^{s1}}\sum_{\substack{l=s_{ik}^p\\ l\in\bar{P}_{ik}}}^{p} u_{ik}^l \leq 1 \qquad \forall m\in M, p\in P \tag{4.7}$$

$$(\lambda_i-1)\sum_{\substack{k\in\Gamma_i^{s1}\\ p\in\mathcal{P}_{ik}}} u_{ik}^p = \sum_{s\in\Psi_i\setminus\{s_1\}}\sum_{\substack{k\in\Gamma_i^{s}\\ p\in\bar{P}_{ik}}} u_{ik}^p \qquad \forall i\in I, p\in P \tag{4.8}$$

$$y_k, v_i, x_{ik}, u_{ik}^p \in \{0,1\} \qquad \forall i\in I, k\in\Gamma_i, p\in\mathcal{P}_{ik}. \tag{4.9}$$

A função objetivo (4.1) busca minimizar o número de equipes utilizadas no planejamento e as penalidades pelas manutenções preventivas não programadas. As restrições (4.2) impõem que uma equipe seja ativada se tiver pelo menos uma manutenção atribuída a ela. As restrições (4.3) garantem que cada manutenção programada seja realizada por equipes que tenham as habilidades para executá-la. As restrições (4.4) garantem que, se uma manutenção for atribuída a uma equipe, então ela será programada nesta equipe. As restrições (4.5) garantem que uma manutenção programada inicie sua execução em apenas um período por habilidade. As restrições (4.6) são responsáveis por construir a programação das manutenções em cada equipe e garantir que as equipes executem apenas uma manutenção por vez. As restrições (4.7), por sua vez, são responsáveis por construir a programação das manutenções em cada máquina e garantir que apenas uma manutenção seja executada por vez em cada máquina. As restrições (4.8) garantem que o período de início de execução de cada manutenção seja igual para todas as equipes a ela atribuídas. As restrições (4.9) especificam o domínio das variáveis.

A formulação $\mathcal{TI}$ é baseada na formulação proposta na Seção 3.4 para o LTPMOSP. Porém, foram adicionadas a ela a variável v_i e as restrições (4.3), (4.5) e (4.8). Além disso, a função objetivo e as restrições (4.5) e (4.7) foram modificadas. Essas modificações foram realizadas para considerar as características do LTPMOSP-MS relativas tanto à exigência de multi-habilidade requerida pelas manutenções, quanto à disponibilidade das equipes. Além disso, o subconjunto $\bar{P}_{ik} \subseteq P$ foi modificado para garantir que as manutenções sejam programadas nas janelas das equipes.

4.3 Heurística de duas fases

A principal ideia da heurística de duas fases, denotada por H2F, é decompor o LTPMOSP-MS em um ou mais subproblemas de menor dimensão, que são resolvidos em duas fases. A primeira fase é responsável por solucionar o problema com parte das restrições do LTPMOSP-MS relaxadas, quais sejam as restrições de não preempção e sobreposição de manutenções no sequenciamento das máquinas. Além disso, não há restrições que garantam que as manutenções com multi-habilidade iniciem em um mesmo instante em todas as equipes em que elas são atribuídas. A solução desse problema atribui as manutenções programadas às equipes, definindo as manutenções que cada equipe deve executar. Essa atribuição é feita de forma a minimizar o número de equipes e o número de manutenções

não executadas, que é o mesmo objetivo do problema original. Assim, a segunda fase tem como objetivo programar as manutenções nas equipes e máquinas, respeitando todas as restrições do problema. É importante destacar que as manutenções que não foram atribuídas a nenhuma equipe na primeira fase não são consideradas na segunda fase. Além disso, pode ocorrer que manutenções atribuídas às equipes na primeira fase não sejam programadas na segunda fase. Após finalizada a segunda fase, tem-se uma solução inteira viável para o LTPMOSP-MS.

Para solucionar o problema relaxado da fase 1, utiliza-se uma formulação matemática, que será denominada de formulação $\mathcal{F}_{\mathcal{I}}$. Ela é uma versão adaptada da eficiente formulação TIP introduzida na Seção 3.5. A formulação $\mathcal{F}_I$ é também baseada em variáveis indexadas no tempo. Nela, o horizonte de planejamento é dividido em partes menores com duração igual. Cada parte, ou melhor, cada período, pode ter uma duração de tempo que possibilita alocar diversas manutenções, ou mesmo, ter uma duração que possibilite alocar uma manutenção em mais de um período. Essa possibilidade de não ter nenhuma condição para definir a duração do período difere a formulação $\mathcal{F}_I$ da formulação $\mathcal{TI}$. Nela, o período pode ter somente a duração igual a ζ, em que esse valor é definido pelos dados das manutenções. Além disso, o número de variáveis e restrições da formulação $\mathcal{TI}$ está fortemente relacionado com o número de períodos e, como ζ é um número pequeno, ele torna a formulação muito grande. Em contrapartida, com a formulação $\mathcal{F}_I$, pode-se ter períodos maiores que geram formulações de dimensões menores. Nesta tese, a formulação $\mathcal{F}_I$ é utilizada apenas considerando períodos com duração muito maior que ζ.

Após finalizada a fase um, é executada a fase dois. A solução da primeira fase possibilita duas estratégias diferentes para a segunda fase. A primeira estratégia consiste em solucionar um subproblema para todo o horizonte de planejamento (H2F-hor). Nessa estratégia, as manutenções atribuídas na fase podem ser programadas em qualquer instante das suas JRs. Diferentemente, a segunda estratégia consiste em dividir o subproblema de todo o horizonte em subproblemas menores e, assim, solucionar cada um deles separadamente e em sequência (H2F-per). Nessa estratégia, cada período da formulação $\mathcal{F}_I$ torna-se um subproblema. Em cada subproblema podem ser programadas apenas as manutenções atribuídas a ele na fase 1. Além disso, as manutenções podem ser programadas apenas nos instantes da JR que estão dentro do subproblema. Desse modo, a solução obtida no final da segunda fase pode ser utilizada para gerar pseudo-cortes que podem ser adicionados na formulação da primeira fase para eliminar a solução corrente. Assim, as duas fases podem ser executadas de forma iterativa (H2F-iter). Portanto, essa tese apresenta três versões para a H2F. Cada uma delas será apresentada e detalhada nas Subseções 4.3.3, 4.3.4 e 4.3.5. Todas as versões da heurística utilizam uma formulação para solucionar o problema relaxado na primeira fase e uma formulação para solucionar um ou mais subproblemas na segunda. Ambas as formulações serão detalhadas nas Subseções 4.3.1 e 4.3.2, respectivamente.

4.3.1 Formulação matemática para a Fase 1 da H2F

A formulação matemática da fase 1 ($\mathcal{F}_{\mathcal{I}}$) é baseada na formulação que combina variáveis indexadas no tempo e de precedência da Seção 3.5. Diferente da formulação citada, a apresentada neste capítulo utiliza apenas as variáveis indexadas no tempo. Além disso, a formulação da Seção 3.5 é utilizada para encontrar uma solução inteira viável para o problema; já a formulação $\mathcal{F}_{\mathcal{I}}$ tem uma abordagem diferente. A nova versão da formulação é utilizada como parte da H2F, com o objetivo de solucionar o problema com relaxação das restrições que garantem a não preempção, a sobreposição no sequenciamento das manutenções nas máquinas e início no mesmo instante de manutenções atribuídas a mais de uma equipe.

Na formulação $\mathcal{F}_I$, o horizonte de planejamento é dividido em períodos com igual duração. Cada período pode ter mais de uma manutenção atribuída a ele. As manutenções podem ser atribuídas a um ou mais períodos, dependendo da duração do período, do tempo de execução e da JR da manutenção.

A formulação $\mathcal{F}_I$ considera o conjunto de períodos $T = \{0, 1, ..., t_{max}\}$. Todos os períodos têm duração definida pelo parâmetro π. Cada período t inicia no instante $t \times \pi$ e finaliza no instante $(t + 1) \times \pi$. Para garantir que as manutenções sejam atribuídas em períodos que pertencem às suas JRs, considere os subconjuntos $\mathcal{T}_{ik}^{(1)}$, $\mathcal{T}_{ik}^{(2)}$ e $\mathcal{T}_{ik}$. Os subconjuntos $\mathcal{T}_{ik}^{(1)}$ e $\mathcal{T}_{ik}^{(2)}$ denotam os períodos em que a manutenção i pode ser atribuída à equipe k, considerando, respectivamente, a primeira e a segunda JR, enquanto $\mathcal{T}_{ik} = \mathcal{T}_{ik}^{(1)} \cup \mathcal{T}_{ik}^{(2)}$. Os parâmetros μ_i^{lb} e μ_i^{ub} são utilizados na formulação $\mathcal{F}_I$ para definir, respectivamente, um limite inferior e superior para o número de períodos em que cada manutenção pode ser atribuída. Esses parâmetros são dados por: $\mu_i^{lb} = \left\lceil \frac{p_i}{\pi} \right\rceil$ e $\mu_i^{ub} = \left\lceil \frac{p_i}{\pi} \right\rceil + 1$.

A formulação $\mathcal{F}_I$ utiliza os conjuntos, os parâmetros apresentados na Seção 4.1.1 e as seguintes variáveis:

(i) y_k é uma variável binária que é igual a 1 se a equipe k é utilizada no planejamento;

(ii) x_{ik} é uma variável binária que é igual a 1 se a manutenção i é executada pela equipe k e 0, caso contrário;

(iii) u_{ikt}^q é uma variável binária que é igual a 1 se a manutenção i é executada pela equipe k no período t na JR q e 0, caso contrário;

(iv) Δ_{ikt}^q é uma variável real não negativa que representa o tempo de execução da manutenção i na equipe k no período t na JR q;

(v) r_{ik}^q é uma variável binária que é igual a 1 se a manutenção i é executada pela a equipe k e está alocada à JR q.

Para que as manutenções sejam alocadas considerando as JRs, são utilizados os parâmetros $t_{ik}^{\alpha_{(1)}}$, $t_{ik}^{\beta_{(1)}}$, $t_{ik}^{\alpha_{(2)}}$, $t_{ik}^{\beta_{(2)}}$, $\delta_{ik}^{\alpha_{(1)}}$, $\delta_{ik}^{\beta_{(1)}}$ $\delta_{ik}^{\alpha_{(2)}}$ e $\delta_{ik}^{\beta_{(2)}}$. Os parâmetros $t_{ik}^{\alpha_{(1)}}$ e $t_{ik}^{\alpha_{(2)}}$ referem-se aos períodos em que inicia a primeira e a segunda JR, respectivamente, e os parâmetros $t_{ik}^{\beta_{(1)}}$ e

$t_{ik}^{\beta_{(2)}}$ aos períodos em que finaliza a primeira e a segunda JR, respectivamente. Eles são dados por $t_{ik}^{\alpha_{(1)}} = \left\lfloor \frac{\alpha_{ik}^{(1)}}{\pi} \right\rfloor$, $t_{ik}^{\alpha_{(2)}} = \left\lfloor \frac{\alpha_{ik}^{(2)}}{\pi} \right\rfloor$, $t_{ik}^{\beta_{(1)}} = \left\lceil \frac{\beta_{ik}^{(1)}}{\pi} \right\rceil - 1$ e $t_{ik}^{\beta_{(2)}} = \left\lceil \frac{\beta_{ik}^{(2)}}{\pi} \right\rceil - 1$. Os parâmetros $\delta_{ik}^{\alpha_{(1)}}$ e $\delta_{ik}^{\alpha_{(2)}}$ são utilizados para calcular o tempo entre o início do período e o início da primeira e da segunda JR, respectivamente, e os parâmetros $\delta_{ik}^{\beta_{(1)}}$ e $\delta_{ik}^{\beta_{(2)}}$, para calcular o tempo entre o início do período e o fim da primeira e da segunda JR. Eles são calculados pelas expressões $\delta_{ik}^{\alpha_{(1)}} = \alpha_{ik}^{(1)} - t_{ik}^{\alpha_{(1)}}\pi$, $\delta_{ik}^{\alpha_{(2)}} = \alpha_{ik}^{(2)} - t_{ik}^{\alpha_{(2)}}\pi$, $\delta_{ik}^{\beta_{(1)}} = \beta_{ik}^{(1)} - t_{ik}^{\beta_{(1)}}\pi$ e $\delta_{ik}^{\beta_{(2)}} = \beta_{ik}^{(2)} - t_{ik}^{\beta_{(2)}}\pi$. Essa transformação dos parâmetros apresentados na Seção 4.1 é necessária devido a divisão do horizonte em períodos. Note que todos os parâmetros dependem do parâmetro π.

Para limitar o tempo nos períodos que iniciam e finalizam as janelas de cada equipe, são considerados os parâmetros $t_k^{\bar{a}_{(2)}}$, $t_k^{\bar{b}_{(1)}}$, $t_k^{\bar{b}_{(2)}}$, $\delta_k^{\bar{a}_{(2)}}$, $\delta_k^{\bar{b}_{(1)}}$ e $\delta_k^{\bar{b}_{(2)}}$. De modo semelhante aos parâmetros da JR, os parâmetros das janelas das equipes são dados pelas expressões: $t_k^{\bar{a}_{(2)}} = \left\lfloor \frac{\bar{a}_{ik}^{(2)}}{\pi} \right\rfloor$, $t_k^{\bar{b}_{(1)}} = \left\lceil \frac{\bar{b}_{ik}^{(1)}}{\pi} \right\rceil - 1$, $t_k^{\bar{b}_{(2)}} = \left\lceil \frac{\bar{b}_{ik}^{(2)}}{\pi} \right\rceil - 1$, $\delta_k^{\bar{a}_{(2)}} = \bar{a}_{ik}^{(2)} - t_{ik}^{\bar{a}_{(2)}}\pi$, $\delta_k^{\bar{b}_{(1)}} = \bar{b}_{ik}^{(1)} - t_{ik}^{\bar{b}_{(1)}}\pi$ e $\delta_k^{\bar{b}_{(2)}} = \bar{b}_{ik}^{(2)} - t_{ik}^{\bar{b}_{(2)}}\pi$.

As JRs das manutenções podem se sobrepor e, com isso, a posição de uma manutenção pode interferir na posição de outra em uma mesma equipe. A Figura 4.3 mostra um exemplo de janelas que se sobrepõem na equipe 1 no período 0 com $\pi = 8$h. As manutenções 1 e 2 possuem, respectivamente, as janelas $[1, 6]$ e $[2, 7]$ e tempos de processamento $p_1 = 3$ e $p_2 = 4$. Na Figura 4.4, os valores das variáveis Δ_{ikt}^q não consideram a sobreposição das janelas, o que permite que as duas manutenções sejam alocadas no período 0 e se sobreponham. Para não permitir isso, pode-se criar uma nova janela utilizando as JRs das manutenções 1 e 2. Essa nova janela pode ser nomeada como janela de sobreposição (JS). A sua duração pode ser utilizada para limitar e relacionar as variáveis Δ_{ikt}^q das manutenções com janelas conflitantes. A Figura 4.4 apresenta a JS gerada pelas JRs das manutenções 1 e 2. Ela inicia na hora 1, que é igual a $\min(1, 2)$, e finaliza na hora 7, que é igual a $\max(6, 7)$. Assim, a duração da JS das JRs das manutenções 1 e 2 é igual a 6, porque $\max(6, 7) - \min(1, 2) = 6$. Como a soma das variáveis Δ_{ikt}^q tem que ser menor que a duração da JS, logo $\Delta_{110}^q + \Delta_{210}^q \leq 6$. Através dessa soma, somente uma das duas manutenções pode ser completamente alocada no período 0. De uma forma geral, a duração da JS das manutenções i e j na equipe k no período t é dada pela expressão: $d_{ijk}^t = \max(\beta_{ik}, \beta_{jk}) - \min(\alpha_{ik}, \alpha_{jk})$. O subconjunto $\Theta_{ik} \subseteq I$ denota as manutenções que têm sobreposição de JR com a manutenção i na equipe k. O subconjunto $\tau_{ijk} \subseteq T$ denota os períodos em que as manutenções i e j possuem sobreposição de janela na equipe k.

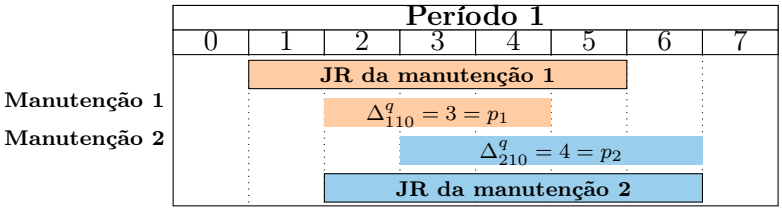


Figura 4.3: Sobreposição das JRs das manutenções 1 e 2.

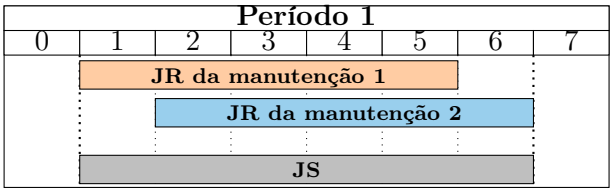


Figura 4.4: JS considerando as janelas das manutenções 1 e 2.

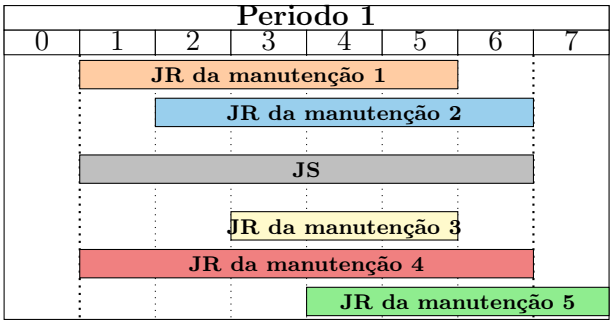


Figura 4.5: Ilustração de uma JS associada às manutenções 1 e 2 e das JRs de cinco manutenções.

Tentando não selecionar manutenções que não podem ser programadas na fase 2, a formulação também considera que podem haver JRs de outros trabalhos dentro do JS das manutenções i e j. Considerando apenas o caso das janelas que iniciam e finalizam dentro da JS, a posição das manutenções nessas janelas interfere na alocação das manutenções i e j e na soma das variáveis Δ. A Figura 4.5 mostra exemplos de JRs que podem ou não se encaixar nesse caso específico. Um exemplo que se encaixa no caso específico é da JR da manutenção 3, que inicia e finaliza dentro da JS. Outro exemplo que pode ser considerado é da JR da manutenção 4, que é exatamente igual à JS. A JR da manutenção 5 não pode ser considerada para esse caso, porque ela finaliza fora da JS. O subconjunto $\theta^t_{ijk} \subseteq I$ denota as manutenções que possuem JRs dentro da JS. Portanto, a seguinte restrição é válida: $\Delta^q_{ikt} + \Delta^q_{jkt} + \sum_{l \in \theta^t_{ijk}} \Delta^q_{lkt} \leq d^t_{ijk}$.

Assim, a formulação $\mathcal{F}_I$ é dada, então, por:

$$\min \quad z_1 = \sum_{k \in K} y_k + \sum_{i \in I} w_i(1 - v_i) \tag{4.10}$$

$$\textbf{s. a:} \quad x_{ik} \leq y_k \qquad \forall i \in I, k \in \Gamma_i \tag{4.11}$$

$$\sum_{k \in \Gamma^s_i} x_{ik} = v_i \qquad \forall i \in I, s \in \Psi_i \tag{4.12}$$

$$\sum_{k \in \Gamma^s_i} \sum_{q \in Q_{ik}} \sum_{t \in \mathcal{T}^q_{ik}} \Delta^q_{ikt} = p_i v_i \qquad \forall i \in I, s \in \Psi_i \tag{4.13}$$

$$\sum_{k \in \Gamma^s_i} \sum_{q \in Q_{ik}} \sum_{t \in \mathcal{T}_{ik}} u^q_{ikt} \leq \mu^{ub}_i v_i \qquad \forall i \in I, s \in \Psi_i \tag{4.14}$$

$$\sum_{k\in\Gamma_i^s}\sum_{q\in Q_{ik}}\sum_{t\in\mathcal{T}_{ik}} u_{ikt}^q \geq \mu_i^{lb} v_i \qquad \forall i\in I, s\in\Psi_i \quad (4.15)$$

$$\Delta_{ikt}^q \leq \pi u_{ikt}^q \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in\mathcal{T}_{ik}^q \quad (4.16)$$

$$\Delta_{ikt}^q \leq \pi x_{ik} \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in\mathcal{T}_{ik}^q \quad (4.17)$$

$$u_{ikt}^q \leq x_{ik} \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in\mathcal{T}_{ik}^q \quad (4.18)$$

$$\sum_{q\in Q_{ik}} r_{ik}^q = x_{ik} \qquad \forall i\in I, k\in\Gamma_i \quad (4.19)$$

$$u_{ikt}^q \leq r_{ik}^q \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in\mathcal{T}_{ik} \quad (4.20)$$

$$\sum_{l=t_1}^{t-1} u_{ikl}^q - t u_{ik(t-1)}^q + t u_{ikt}^q < t \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in\mathcal{T}_{ik}^q\setminus\{t_1\} \quad (4.21)$$

$$\sum_{i\in\Omega_k}\sum_{\substack{q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} \Delta_{ikt}^q \leq \pi \qquad k\in K, t\in T \quad (4.22)$$

$$\sum_{i\in\Phi_m}\sum_{k\in\Gamma_i^{s1}}\sum_{\substack{q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} \Delta_{ikt}^q \leq \pi \qquad m\in M, t\in T \quad (4.23)$$

$$\sum_{k\in\Gamma_i^{s1}}\sum_{\substack{q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} \Delta_{ikt}^q = \sum_{\bar{k}\in\Gamma_i^s}\sum_{\substack{q\in Q_{i\bar{k}},\\ t\in\mathcal{T}_{i\bar{k}}^q}} \Delta_{i\bar{k}t}^q \qquad i\in I, s\in\Psi_i\setminus\{s_1\}, t\in T \quad (4.24)$$

$$(\lambda_i - 1)\sum_{k\in\Gamma_i^{s1}}\sum_{\substack{q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} u_{ik}^t = \sum_{s\in\Psi_i\setminus\{s_1\}}\sum_{k\in\Gamma_i^{si}}\sum_{\substack{q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} u_{ik}^t \qquad \forall i\in I, t\in T \quad (4.25)$$

$$\Delta_{ikt_{ik}^{\alpha_q}}^q \leq \pi - \delta_{ik}^{\alpha_q} r_{ik}^q \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik} \quad (4.26)$$

$$\Delta_{ikt_{ik}^{\beta_q}}^q \leq \delta_{ik}^{\beta_q} r_{ik}^q + \pi(1 - r_{ik}^q) \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik} \quad (4.27)$$

$$\sum_{\substack{i\in\Omega_k,\\ t\in\mathcal{T}_{ik}^{(2)}}} \Delta_{ikt}^{(2)} \leq \pi - \bar{\delta}_k^t \qquad t = t_k^{\bar{a}_{(2)}}, \forall k\in K \quad (4.28)$$

$$\sum_{\substack{i\in\Omega_k,\\ q\in Q_{ik},\\ t\in\mathcal{T}_{ik}^q}} \Delta_{ikt}^q \leq \bar{\delta}_k^t \qquad k\in K, q\in\mathcal{Q}, t = t_k^{\mathcal{B}_q} \quad (4.29)$$

$$\Delta_{ikt}^q + \Delta_{jkt}^q + \sum_{l\in\theta_{ijkt}^q} \Delta_{lkt}^q \leq d_{ijkt}^q \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, \forall j\in\Theta_{ik}^q, t\in\tau_{ijk}^q \quad (4.30)$$

$$y_k \in \{0,1\} \qquad \forall k\in K \quad (4.31)$$

$$x_{ik} \in \{0,1\} \qquad \forall i\in I, k\in\Gamma_i \quad (4.32)$$

$$u_{ikt}^q \in \{0,1\} \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in T \quad (4.33)$$

$$0 \leq \Delta_{ikt}^q \leq \pi \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik}, t\in T \quad (4.34)$$

$$r_{ik}^q \in \{0,1\} \qquad \forall i\in I, k\in\Gamma_i, q\in Q_{ik} \quad (4.35)$$

A função objetivo (4.10) busca minimizar o número de equipes ativadas e as pena-

lidades pelas manutenções não programadas. As restrições (4.11) impõem que uma equipe é ativada somente se tiver pelo menos uma manutenção atribuída a ela. As restrições (4.12) garantem que cada manutenção seja realizada por equipes que tenham as habilidades para executá-la. Além disso, elas garantem que a manutenção seja atribuída apenas a uma equipe para cada habilidade que ela possui. As restrições (4.13) garantem que a soma dos tempos de processamento nos períodos (Δ_{ik}^{t}) de uma manutenção seja igual ao tempo de processamento da mesma. As restrições (4.14) e (4.15) definem um limite superior e inferior, respectivamente, de número de períodos que uma manutenção pode ser atribuída. As restrições (4.16) impõem que as variáveis Δ_{ik}^{t} sejam diferentes de zero apenas nos períodos e equipes em que a manutenção é atribuída. As restrições (4.17) e (4.18) garantem que cada manutenção é atribuída a períodos apenas na equipe em que é programada. As restrições (4.19) garantem que a manutenção é alocada à apenas uma JR. As restrições (4.20) garantem que a manutenção é atribuída apenas nos períodos da JR na qual é atribuída a ela. As restrições (4.21) garantem que os períodos são atribuídos consecutivamente para as manutenções programadas. As restrições (4.22) e (4.23) garantem que o tempo de processamento das manutenções (Δ_{ik}^{t}) em um período seja menor ou igual a π em cada equipe e máquina, respectivamente. As restrições (4.24) garantem que a duração do tempo de processamento de cada manutenção em cada período seja igual em todas as equipes atribuídas a ela. As restrições (4.25) garantem que cada manutenção seja atribuída em períodos iguais em todas as equipes à ela atribuídas. As restrições (4.26)) e (4.27) limitam o valor das variáveis Δ_{ik}^{t} nos períodos de início e fim, respectivamente, das JRs. As restrições (4.28) e (4.29) limitam a duração dos períodos em que iniciam e finalizam as janelas das equipes. As restrições (4.30) associam as manutenções que possuem sobreposição de JRs e limitam o valor das variáveis Δ_{ik}^{t} para evitar que se tenha sobreposição de manutenções.

Em comparação com a formulação introduzida na Seção 3.5, deve-se observar que há três situações diferentes envolvendo o conjunto de restrições da formulação $\mathcal{F}_I$. As restrições (4.11), (4.16), (4.17), (4.18) e (4.22) estão presentes na formulação apresentada na Seção 3.5 e não foram modificadas. As restrições (4.13), (4.14), (4.15), (4.21) e (4.23), por sua vez, também estão presentes na formulação postas na Seção 3.5, mas foram modificadas. Por fim, as restrições (4.12), (4.19), (4.20), (4.24), (4.25), (4.26), (4.27), (4.28), (4.29) e (4.30) são estruturas novas, não existentes na formulação da Seção 3.5.

As variáveis e as restrições da formulação $\mathcal{F}_I$ mostram forte dependência quanto ao número de períodos, que, contudo, depende do valor de π. Portanto, o valor de π interfere fortemente nas dimensões da formulação, bem como, obviamente, na solução gerada por ela. O impacto de π na solução é analisado na Seção 4.4

4.3.2 Formulação matemática para a fase 2 da H2F

A solução de cada subproblema da segunda fase é encontrada através de uma formulação baseada na formulação $\mathcal{TI}$, que será denotada por $\mathcal{F}_{\mathcal{II}}$. Esta formulação considera apenas as manutenções, equipes e períodos associados ao respectivo subproblema.

Seja $\mathcal{S}$ o conjunto de subproblemas, que está associado a como o horizonte de planejamento é dividido. Nos algoritmos H2F-hor e H2F-iter, mostrados nos Algoritmos 8 e 9, respectivamente, tem-se um único subproblema que considera todo o horizonte; assim, $|\mathcal{S}| = 1$. No Algoritmo H2F-per, para cada período $t \in T$ da formulação $\mathcal{F}_{\mathcal{I}}$, há um subproblema. Neste caso, $|\mathcal{S}| = |T|$. A partir da solução encontrada na primeira fase, tem-se os subconjuntos I_{sub}, K_{sub} e $\bar{I}^k_{sub}$. O subconjunto $I_{sub} \subseteq I$ contém as manutenções do subproblema *sub*. Sejam $\bar{v}$ e $\bar{u}$ valores das variáveis de decisão obtidos na fase 1; então, $I_{sub} = \{i \in I : \bar{v}_i = 1\}$ para os algoritmos H2F-hor e H2F-ite, enquanto que $I_{sub} = \{i \in I : \bar{u}^q_{ikt} = 1$ para algum k,$q\}$ para o algoritmo H2F-iter, em que t corresponde a *sub*. Por exemplo, uma manutenção atribuída ao período 1 na solução da primeira fase é atribuída ao subproblema 1. O subconjunto $K_{sub} \subseteq K$ denota as equipes às quais alguma manutenção foi alocada na fase 1 no período $t \in T$ correspondente a $sub \in \mathcal{S}$. O subconjunto $M_{sub} \subseteq M$ denota as máquinas associadas às manutenções em I_{sub}. O subconjunto $\bar{I}^k_{sub}$ denota o subconjunto de manutenções atribuídas à equipe k na fase I no período t correspondente a *sub*. Além disso, se $i \in I_{sub}$, então a manutenção i foi alocada na primeira fase do algoritmo à equipe k^s_i para cada habilidade $s \subseteq S_i$ que ela requer. Por fim, o subconjunto $\mathcal{T}_{sub}$ denota o conjunto de períodos da formulação $\mathcal{F}_{\mathcal{II}}$ para o subproblema $sub \in \mathcal{S}$.

A formulação $\mathcal{F}_{\mathcal{II}}$ utiliza as seguintes variáveis binárias:

- y_k, que é igual a 1 se a equipe k é utilizada no planejamento e 0, caso contrário, com $i \in I$;
- v_i, que é igual a 1 se a ordem i é executada e 0, caso contrário, com $i \in I$;
- u^t_{ik}, que é igual a 1 se a manutenção i inicia sua execução no período t com a equipe k e 0, caso contrário.

Então, a formulação $\mathcal{F}_{\mathcal{II}}$ é dado por:

$$\min \quad z_2 = \sum_{k \in K_{sub}} y_k + \sum_{i \in I_{sub}} w_i \left(1 - v_i\right) \tag{4.36}$$

$$\textbf{suj. a:} \quad v_i \leq y_k \qquad \forall k \in K_{sub}, i \in \bar{I}^k_{sub} \tag{4.37}$$

$$\sum_{t \in \mathcal{T}_{ik}} u^t_{ik^s_i} = v_i \qquad \forall i \in I_{sub}, s \in \Psi_i \tag{4.38}$$

$$\sum_{i \in \bar{I}^k_{sub}} \sum_{\substack{l=s^t_{ik} \\ l \in \mathcal{T}_{ik}}}^{t} u^l_{ik} \leq 1 \qquad \forall k \in K_{sub}, t \in T_{sub} \qquad (4.39)$$

$$\sum_{i \in \Phi_m \cap I_{sub}} \sum_{\substack{l=h^t_{ik^{s_1}_i} \\ l \in \mathcal{T}_{ik^{s_1}_i}}}^{t} u^l_{ik^{s_1}_i} \leq 1 \qquad \forall m \in M_{sub}, t \in T_{sub} \qquad (4.40)$$

$$(\lambda_i - 1) u^t_{ik^{s_1}_i} = \sum_{s \in \Psi_i \setminus \{s_1\}, t \in \mathcal{T}_{ik^s_i}} u^t_{ik^s_i} \qquad \forall i \in I_{sub}, t \in \mathcal{T}_{ik^s_i} \qquad (4.41)$$

$$y_k \in \{0,1\} \qquad \forall k \in K \qquad (4.42)$$

$$v_i \in \{0,1\} \qquad \forall i \in I \qquad (4.43)$$

$$u^t_{ik} \in \{0,1\} \qquad \forall i \in I, k \in \Gamma_i, t \in T \qquad (4.44)$$

A função objetivo (4.36) e as restrições (4.38), (4.39), (4.40), (4.41) são semelhantes às restrições da formulação $\mathcal{TI}$, tendo, como diferença, apenas os conjuntos e subconjuntos. As restrições (4.37) são uma versão modificada das restrições (4.2); porém, ambas impõem que uma equipe é ativada somente se tiver pelo menos uma manutenção atribuída a ela. Note que a única variável que está presente na formulação $\mathcal{TI}$ e não na formulação $\mathcal{F}_{\mathcal{II}}$ é a variável x_{ik}. Na formulação $\mathcal{TI}$, ela é utilizada para atribuir as manutenções às equipes. Com isso definido na primeira fase, não é necessário utilizar a variável x_{ik} na formulação $\mathcal{F}_{\mathcal{II}}$.

4.3.3 Heurística de duas fases com decomposição do problema na segunda fase

A solução dada pela formulação $\mathcal{F}_I$ atribui as manutenções às equipes e aos períodos. Essa solução possibilita a decomposição do problema da fase 2 em um subproblema para cada período. A Heurística 2PH-per utiliza na segunda fase a estratégia de resolver cada subproblema separadamente e em sequência. O Algoritmo 7 apresenta o pseudocódigo da 2PH-per. Esse algoritmo utiliza o conjunto $\mathcal{S}$, que denota os subproblemas. Note que, para cada período $t \in T$, tem-se um subproblema $sub \in \mathcal{S}$.

O Algoritmo 7 recebe, como entrada, os parâmetros ti_1 e TL, que são o tempo limite de execução da primeira fase e da própria H2F-per. Na linha 3, o TL é atualizado subtraindo o tempo de execução da primeira fase ($tempo_{S^1}$) do tempo limite da matheurística inicial. O parâmetro ti_2 é o tempo limite de execução de cada subproblema. Ele é dado pela divisão do tempo restante da matheurística pelo número de subproblemas que faltam ser solucionados (linha 4). Esse valor é atualizado a cada solução de um subproblema para que a matheurística não passe do

Algoritmo 7: H2F-per

Entradas: ti_1, TL
Saída : solução S

1 $S^1 \leftarrow resolveProblemaRelaxado(ti_1)$
2 $TL \leftarrow TL - tempo_{S^1}$
3 **for** $sub = 0$; $sub < |\mathcal{S}|$; $sub++$ **do**
4 $\quad ti_2 = \dfrac{TL}{|\mathcal{S}| - sub}$
5 $\quad S^2_{sub} \leftarrow resolverSubproblemaPeriodos(S^1_{sub}, ti_2)$
6 $\quad S = S \cup S^2_{sub}$
7 $\quad TL \leftarrow TL - tempo_{S^2}$
8 **end**
9 **return** S

tempo limite de execução. A linha 1 do Algoritmo 7 é a primeira fase da H2F-per. Nela é resolvida o LTPMOSP-MS relaxado. A solução obtida na primeira fase (S^1) é dividida para cada subproblema $sub \in \mathcal{S}$ ($S^1 = S^1_{sub} \cup ...S^1_{|\mathcal{S}|}$). Cada S^1_{sub} é utilizada para solucionar cada subproblema sub da segunda fase (linha 5). O *loop* entre as linhas 3 e 8 refere-se à segunda fase da H2F-per e ilustra a resolução separada e em sequência de cada subproblema. Após a execução de cada subproblema, tem-se a solução S^2_{sub} que se une à solução S (linha 6). Ao final da fase 2, S é uma solução inteira factível para o LTPMOSP-MS.

Como os subproblemas da 2PH-per são solucionados separadamente, cada subproblema tem uma formulação $\mathcal{F}_{\mathcal{II}}$. É importante destacar que a formulação $\mathcal{F}_{\mathcal{II}}$ para cada subproblema utilizará os subconjuntos apenas com as manutenções, equipes e períodos associados ao respectivo subproblema.

Um detalhe importante é que uma manutenção pode ser incluída em mais de um subproblema. Por exemplo, uma manutenção é incluída nos subproblemas 1 e 2. Ela pode ser programada na solução do subproblema 1. Porém, caso ela não seja, ela é adicionada ao subproblema 2, junto às demais manutenções desse subproblema e pode ser, portanto, programada.

4.3.4 Heurística de duas fases com um único problema na segunda fase

Diferentemente da H2F-per, a H2F-hor soluciona um único subproblema na segunda fase. Esse subproblema considera todo o horizonte de planejamento e todas as manutenções que foram atribuídas a equipes na primeira fase. Vale destacar que, na H2F-hor, as manutenções são livres para serem programadas em qualquer instante de tempo da sua JR, enquanto que na H2F-per as manutenções podem ser progra-

madas somente no intervalo de tempo do subproblema em que foram atribuídas. O Algoritmo 8 apresenta o pseudocódigo da H2F-hor.

Algoritmo 8: H2F-hor

Entradas: ti_1, TL
Saída : solução S

1 $S^1 \leftarrow resolveProblemaRelaxado(ti_1)$
2 $ti_2 \leftarrow TL - time_{S^1}$
3 $S \leftarrow resolverSubproblemaHorizonte(S^1, ti_2)$
4 **return** S

A H2F-hor também tem como os parâmetros de entrada ti_1 e TL. Eles apresentam a mesma função que na H2F-per. O parâmetro ti_2 também está presente e limita o tempo de execução do único subproblema da fase 2. A linha 1 do Algoritmo 8 é a primeira fase do H2F-hor. O subproblema da segunda fase recebe a solução da fase 1 (S^1) e gera uma solução inteira factível para o LTPMOSP-MS (S).

Como a H2F-hor tem um único subproblema na segunda fase, ela terá apenas uma formulação $\mathcal{F}_{\mathcal{II}}$. Assim, os subconjuntos consideram todas as manutenções atribuídas a alguma equipe na primeira fase e todos os períodos do horizonte de planejamento.

4.3.5 Heurística de duas fases iterativa

A H2F-iterativa é a versão iterativa da H2F-hor. A solução gerada na segunda fase da H2F pode ter manutenções não programadas. Para forçar a mudança de equipe das manutenções que não foram programadas e ter uma solução diferente na primeira fase, é adicionado um pseudo-corte na formulação $\mathcal{F}_I$. Os pseudo-cortes consistem em desigualdades lineares que são geradas com o propósito de excluir estrategicamente algumas soluções do problema. O prefixo pseudo refere-se a que essas desigualdades podem não ser válidas para garantir que pelo menos uma solução globalmente ótima será retirada (Glover et al., 2011).

Sejam $\bar{s} = (\bar{y}, \bar{x}, \bar{u}, \bar{\Delta}, \bar{r})$ uma lista de valores para as variáveis da primeira fase em uma certa iteração; o subconjunto $I^{F_{II}} \subseteq I$, que denota as manutenções não programadas na solução da fase 2; o conjunto $\Upsilon = \{(i,k) \in I \times \Gamma_i \text{ tal que } \bar{x}_{ik} = 1\}$; e o conjunto $\Upsilon^c = \{(i,k) \in I \times \Gamma_i \text{ tal que } \bar{x}_{ik} = 0\}$. Então, para eliminar a solução $\bar{s}$ do espaço de busca da primeira fase, é adicionado o seguinte pseudo-corte:

$$\sum_{k:(i,k)\in\Upsilon} x_{ik} \leq |\Psi_i| - 1 \qquad \forall i \in I^{F_{II}} \tag{4.45}$$

As restrições (4.45) impõem que, caso a manutenção i seja programada na solução da primeira fase, ela, então, é atribuída a pelo menos uma equipe diferente daquela

a que ela foi atribuída na solução da primeira fase. É importante destacar que esse pseudo-corte não impõe que as manutenções do subconjunto $I^{F_{II}} \subseteq I$ sejam programadas. A escolha se a manutenção é ou não programada é dada pela solução da formulação. A apresentação da H2F-iterativa e seu funcionamento com o pseudo-corte podem ser vistos no pseudocódigo no Algoritmo 9.

Algoritmo 9: H2F-iter

Entradas: ti_1, TL, TL_2
Saída : solução S

```
1  end ← false
2  while end = false do
3  |   S^1 ← resolveProblemaRelaxado(ti_1)
4  |   ti_2 ← min(TL − tempo_{S^1}, TL_2)
5  |   S ← resolverSubproblemaHorizonte(S^1, ti_2)
6  |   TL ← TL − tempo_{S^2}
7  |   if Todas manutenções foram programadas or TL ≤ 0 then
8  |   |   end ← true
9  |   end
10 |   else
11 |   |   Remove os pseudo-cortes (4.45) da iteração anterior, caso tenha
12 |   |   Adiciona os novos pseudo-cortes
13 |   end
14 end
15 Return S
```

Do mesmo modo que as outras versões do H2F, a H2F-iter também utiliza os parâmetros ti_1, TL e ti_2. No entanto, somente a H2F-iter utiliza o parâmetro TL_2. Eventualmente, o *solver* pode necessitar de um tempo maior ou igual a TL para solucionar a formulação da segunda fase. Caso isso aconteça, H2F-iter terá apenas uma iteração e, assim, não utilizará os benefícios do pseudo-corte. Para evitar que isto ocorra, é utilizado o parâmetro TL_2, que limita o valor do parâmetro ti_2. O valor do parâmetro ti_2 é atualizado a cada iteração. Cada execução do *loop* entre as linhas 2 e 13 do Algoritmo 9 representa uma iteração da H2F-iterativa. A primeira fase de cada iteração é executada na linha 3, enquanto que a segunda fase é executada na linha 5. Nas linhas 4 e 6 os valores dos parâmetros TL e ti_2 são atualizados a cada iteração. Se o valor de TL for menor ou igual a zero, ou todas as manutenções atribuídas na primeira fase sejam programadas na solução da formulação $\mathcal{F}_{\mathcal{II}}$ (linha 7), o parâmetro *end* recebe *true* (linha 8) e a H2F-iter é finalizada. No entanto, caso não aconteça nenhuma das duas condições, são removidos os pseudo-cortes da iteração e adicionados os novos pseudo-cortes (linhas 11 e 12). Eles são gerados a partir da solução encontrada na segunda fase da respectiva iteração. Com os pseudo-cortes adicionados na formulação $\mathcal{F}_I$, é executada uma nova iteração.

4.4 Experimentos Computacionais para a solução do LTPMOSP-MS

Esta seção apresenta os resultados computacionais obtidos com a resolução do LTPMOSP-MS utilizando a formulação $\mathcal{TI}$ e as três versões da H2F. Para realizar os experimentos, foram geradas instâncias utilizando as de Aquino et al. (2018a), com adaptações que permitem manutenções com multi-habilidades e janelas nas equipes. Essas instâncias foram divididas em dois conjuntos: escala pequena ($\mathcal{P}$) e escala média ($\mathcal{M}$). Essa divisão é baseada no número de manutenções disponíveis no planejamento e na divisão proposta por Aquino et al. (2018a). As Subseções 4.4.1 e 4.4.2 apresentam e detalham as instâncias e os resultados computacionais obtidos para cada conjunto, respectivamente.

Todos os experimentos apresentados nessa seção foram executados em um computador equipado com um processador Intel i3-10100T 3.00 GHz, 16 GB de RAM e sistema operacional Ubuntu 18.04.6 LTS. Os códigos foram implementados na linguagem `C++` utilizando a biblioteca *Concert Technology* e todas as formulações foram solucionadas via solver CPLEX 12.9.

4.4.1 Experimentos do conjunto de instâncias $\mathcal{P}$

A seguir é detalhado o procedimento de geração das instâncias do conjunto $\mathcal{P}$ e as características dessas instâncias. Em seguida, é apresentado e analisado os resultados de um experimento computacional com as respectivas instâncias.

4.4.1.1 Geração de instâncias para o conjunto $\mathcal{P}$

As instâncias do conjunto $\mathcal{P}$ têm entre 20 e 80 manutenções. As instâncias desse conjunto foram geradas utilizando como base as instâncias apresentadas em Aquino et al. (2018a). Assim, a maioria dos dados das manutenções foi mantida. A única parte alterada foi o número de habilidades necessárias para executar cada manutenção. A habilidade associada à manutenção $i \in I$ da instância original está no subconjunto Ψ_i e ela foi mantida. As demais habilidades que pertencem a esse subconjunto foram adicionadas aleatoriamente. Para escolher quais as habilidades serão adicionadas, foi considerado o subconjunto μ_m de habilidades para cada máquina $m \in M$. As novas habilidades adicionadas para cada manutenção i são limitadas àquelas que pertencem ao subconjunto μ_{e_i}. Já o número total de habilidades para cada manutenção i é definido aleatoriamente entre 0 e $(|\mu_{e_i}| - 1)$. Para verificar o impacto nos experimentos, o número de manutenções com multi-habilidade em cada instância é dado pelo produto $\Omega \times |I|$. O parâmetro Ω significa a porcentagem de

manutenções com multi-habilidade. Nos experimentos desta tese foram utilizados valores de Ω iguais a 25%, 50% e 80%. Para cada valor de Ω, foram geradas duas instâncias.

Para gerar instâncias com janelas nas equipes, foi definida inicialmente a duração do intervalo que separa as duas janelas. A duração do intervalo para cada equipe k é dada pelo produto $\Phi \times h_k$, em que Φ pode ser igual a $0,05$ ou $0,15$, e h_k representa a disponibilidade da equipe segundo apresentada no trabalho de Aquino et al. (2018a). Com a duração do intervalo definida, o próximo passo é definir o seu início. Esse início é gerado aleatoriamente. Através do intervalo, é definido o final da primeira janela ($\bar{b}_k^{(1)}$) e o início da segunda janela ($\bar{a}_k$). O início da primeira janela é igual a zero e o final da segunda janela ($b_k^{(2)}$) é igual ao parâmetro h_k de Aquino et al. (2018a). Para cada valor Φ, foram geradas duas instâncias.

Portanto, para cada instância de Aquino et al. (2018a), foram geradas 24 ($2 \times 2 \times 2 \times 3$) novas instâncias. As instâncias utilizadas do artigo citado possuem $|I| = 20$, 40 e 80, $|K| = 2$, 3, 4 e 5 e $|M| = 2$ e 4. Ao todo, foram adaptadas 18 instâncias originais do trabalho de Aquino et al. (2018a) e, com elas, foram geradas 432 instâncias (18×24). Todas essas instâncias estão disponíveis em `https://github.com/joaoluizandrade/LTPMOSP-MS/tree/main/small`.

4.4.1.2 Resultados dos experimentos computacionais

Esta seção compara os resultados da formulação $\mathcal{TI}$ com as três versões da H2F com diferentes valores de π na formulação da primeira fase. Para cada versão da H2F, foram realizados experimentos com os seguintes valores do parâmetro π: 24, 48, 72 e 96. O valor desse parâmetro está associado ao número de horas. Logo, para $\pi = 24$, os períodos têm a duração de 24 horas ou equivalente a um dia. Desse modo, os experimentos foram realizados com valores de π que equivalem a um, dois, três e quatro dias, respectivamente. Para facilitar a apresentação, o valor de π é representado da seguinte forma nas diferentes versões da matheurística: H2F-per$_\pi$, H2F-hor$_\pi$ e H2F-ite$_\pi$. Como os parâmetros de entrada das manutenções e das equipes são fracionários, cada período das formulações $\mathcal{TI}$ e $\mathcal{F}_{\mathcal{II}}$ tem a duração de 0,1 hora.

Para os experimentos com a formulação $\mathcal{TI}$, o tempo limite de execução do *solver* CPLEX foi definido para 3.600 segundos; para as diferentes versões de H2F, o tempo limite de execução, em segundos, é igual ao número de manutenções disponíveis. Para se ter esses valores de tempo limite de execução nas heurísticas, os parâmetros de entrada associados ao tempo de execução têm os valores iguais a $ti_1 = |I|$, $TL = |I|$ e $TL_2 = \frac{|I|}{2}$ segundos. Note que os tempos ti_1 e TL são iguais. Isso foi assim definido porque o tempo de execução que o *solver* requer para solucionar a formulação $\mathcal{F}_I$ é

relativamente baixo. O parâmetro ti_1 desempenha um papel mais importante apenas nos experimentos do conjunto $\mathcal{M}$.

A Tabela 4.3 apresenta e compara os resultados computacionais da formulação $\mathcal{TI}$ com as diferentes versões da H2F com quatro valores de π. Cada linha está associada a um método de solução e indica o valor médio ou percentual considerando todas as instâncias do conjunto $\mathcal{P}$. A coluna $gap_1(\%)$ indica o valor médio do *gap* entre o melhor limite inferior (lb) e o melhor limite superior (ub) encontrado durante toda a execução do método de solução. Para a formulação $\mathcal{TI}$, estes valores correspondem aos encontrados no final de execução do *solver*. Para a H2F, o valor do limite inferior usado é a solução encontrada pela formulação $\mathcal{F}_I$ na primeira ou única iteração, enquanto o valor de limite superior é a solução obtida no final de execução da respectiva heurística. O valor de gap_1 é dado por:

$$gap_1 = 100 \times \frac{ub - lb}{ub}$$

A coluna $\%opt_1$ indica o percentual de instâncias do conjunto $\mathcal{P}$ para as quais $gap_1 = 0$. A coluna $gap_2(\%)$ apresenta valores apenas para H2F. Isso ocorre porque o limite inferior (lb_{best}) utilizado para calcular o gap_2 é o melhor valor encontrado pela formulação $\mathcal{F}_I$ dentre os quatro valores de π, enquanto o valor de limite superior é a solução obtida no final de execução da respectiva heurística. O valor de gap_2 é dado por:

$$gap_2 = 100 \times \frac{ub - lb_{best}}{ub}$$

A coluna $\%opt_2$ indica o percentual de instâncias do conjunto $\mathcal{P}$ para as quais $gap_2 = 0$. A coluna RPD(%) mostra o *Relative Percentage Deviation*(RPD). Esse indicador compara o limite superior encontrado pela formulação $\mathcal{TI}$ (ub_{FIT}) com o limite superior encontrado pela respectiva H2F (ub_{2PH}). O valor de RPD é dado por:

$$RPD = 100 \times \frac{ub_{2PH} - ub_{FIT}}{ub_{2PH}}$$

Assim, $RPD > 0$ indica que a formulação $\mathcal{TI}$ encontrou soluções melhores que as determinadas pela respectiva H2F. Se $RPD < 0$, então as soluções encontradas pela respectiva H2F são melhores que as encontradas pela formulação $\mathcal{TI}$. A coluna $time(s)$ indica o valor médio do tempo de execução do método de solução. A coluna $t_{FI}(s)$ indica o valor médio de tempo de execução da formulação $\mathcal{F}_I$ da primeira ou única iteração.

Os resultados da coluna opt_1 mostram que a formulação $\mathcal{TI}$ conseguiu encontrar um maior percentual de soluções ótimas do que a H2F. No entanto, a H2F-ite$_{24}$ conseguiu encontrar um valor percentual muito próximo e obteve um valor médio de

Tabela 4.3: Resultados computacionais para a formulação $\mathcal{TI}$ e para as diferentes versões da H2F para solucionar instâncias do conjunto $\mathcal{P}$.

	$\mathbf{gap_1(\%)}$	$\mathbf{\%opt_1}$	$\mathbf{gap_2(\%)}$	$\mathbf{\%opt_2}$	**RPD(%)**	**time(s)**	$\mathbf{t_{FI}(s)}$
Form. $\mathcal{TI}$	6,97	**81,71**	**-**	**-**	**-**	806,30	**-**
H2F-per$_{24}$	30,95	15,97	30,95	15,97	1,57	**2,47**	0,94
H2F-per$_{48}$	43,46	12,96	27,77	21,76	-20,40	6,27	0,42
H2F-per$_{72}$	42,63	12,27	25,29	25,00	-34,87	12,11	0,14
H2F-per$_{96}$	41,77	16,20	21,83	30,79	-33,34	17,89	**0,06**
H2F-hor$_{24}$	7,10	72,69	7,10	73,38	-75,65	16,29	0,95
H2F-hor$_{48}$	35,45	34,03	9,84	64,12	-83,37	16,24	0,42
H2F-hor$_{72}$	35,72	30,32	10,35	63,66	-67,95	16,24	0,13
H2F-hor$_{96}$	36,07	30,32	10,60	60,42	-72,41	15,66	**0,06**
H2F-ite$_{24}$	**5,02**	78,47	**5,02**	**79,17**	-77,80	22,71	0,93
H2F-ite$_{48}$	33,40	38,89	6,84	74,07	**-86,93**	35,81	0,41
H2F-ite$_{72}$	34,11	34,03	7,74	72,92	-70,71	36,48	0,14
H2F-ite$_{96}$	33,75	34,95	7,31	71,06	-76,07	36,31	**0,06**

gap_1 menor do que a formulação $\mathcal{TI}$, em um tempo de execução significativamente menor. O menor valor de gap_1 indica que a H2F-ite$_{24}$ conseguiu encontrar melhores limites inferiores e superiores. É importante destacar que esses bons valores de gap_1 são encontrados devido aos bons limites inferiores gerados pela formulação $\mathcal{F}_{\mathcal{I}}$. A H2F-hor$_{24}$ obteve valores de gap_1 e opt_1 que são aproximadamente tão bons quanto a H2F-ite$_{24}$. As melhores soluções geradas por H2F-ite$_{24}$ em relação a H2F-hor$_{24}$ mostram eficiência do processo iterativo e do pseudo-corte. Esse padrão de melhora de solução se estende para todos os valores de π da H2F-hor$_\pi$ e H2F-ite$_\pi$.

Comparando os resultados das colunas gap_1 e opt_1 da H2F para os quatro valores de π, nota-se uma diferença entre eles. Isso ocorre porque a formulação $\mathcal{FF}_I$ gera limites inferiores diferentes para cada valor de π. Para as três versões da heurística, verifica-se que a H2F com $\pi = 24$ obteve os menores valores de gap_1. Isso está associado aos bons limites inferiores que a formulação $\mathcal{F}_I$ com $\pi = 24$ consegue gerar. Quando os limites superiores são comparados com limites inferiores fixos na coluna gap_2, verifica-se uma redução significativa do valor médio do gap_2 e um aumento do percentual de opt_2 em relação a gap_1 e opt_1 da H2F. Esses valores mostram que os limites superiores encontrados por H2F com $\pi = 48$, 72 e 96 apresentam qualidade próxima dos encontrados pela H2F com $\pi = 24$.

Os limites superiores gerados pela H2F e a formulação $\mathcal{TI}$ são comparados na coluna $RPD(\%)$. Os valores negativos mostram que a maioria das versões da H2F conseguiu encontrar limites superiores melhores do que a formulação $\mathcal{TI}$. Esses melhores limites superiores se concentram principalmente nas instâncias de maior

escala do conjunto $\mathcal{P}$, que possuem 80 manutenções.

Com relação aos tempos de execução dos métodos de solução, a formulação $\mathcal{TI}$ apresenta um valor médio significativamente superior aos encontrados por H2F. Quando comparada às versões da heurística, a H2F-per apresenta os menores tempos. Esses baixos tempos estão associados à divisão do horizonte em subproblemas menores e à resolução deles separadamente. Os valores médios de tempo da H2F-hor são maiores que os encontrados pela H2F-per, porque estão vinculados à solução de um único subproblema na segunda fase da H2F. Como esperado, os tempos da H2F-ite são maiores que os da H2F-hor, uma vez que uma é a versão iterativa da outra. Por fim, a comparação entre os valores médios de tempo de execução mostrados nas colunas $time(s)$ e t_{F1} destaca a diferença de tempo de cada fase da H2F.

A Tabela 4.4 apresenta resultados de tempo de execução dos métodos de solução e porcentagem de manutenções programadas para diferentes valores de $|I|$ e Ω. A tabela está dividida em duas partes. A primeira parte apresenta os valores médios de tempo de execução, enquanto a segunda parte apresenta os valores percentuais de manutenções programadas. Cada linha da tabela mostra os valores médios ou o percentual para um conjunto de instâncias que possuem a configuração apresentada nas duas primeiras linhas da tabela. As colunas $|I|$ e Ω indicam, respectivamente, o número de manutenções disponíveis para o planejamento e o percentual de manutenções com multi-habilidades nas instâncias agrupadas. As demais colunas da Tabela 4.4 referem-se aos métodos de solução e aos respectivos valores de π utilizados na versão da H2F.

Os resultados associados aos valores de tempo computacional na Tabela 4.4 mostram um crescimento dos valores médios conforme aumenta o número de manutenções nas instâncias. A diferença de tempo entre as instâncias com $|I| = 20$ e $|I| = 80$ é expressiva, tanto nas diferentes versões da H2F, bem como na formulação $\mathcal{TI}$. Esse crescimento nos valores médios de tempo também ocorre com o aumento do percentual de manutenções com multi-habilidade, principalmente nos valores da formulação $\mathcal{TI}$.

Com relação ao percentual de manutenções programadas, nota-se que os valores se reduzem conforme o valor de Ω aumenta. Isso ocorre porque as manutenções com multi-habilidade requerem mais de uma equipe. Logo, quando alocadas, tem-se menor tempo disponível para programar outras manutenções. Quando comparados, os valores médios percentuais entre os métodos de solução mostram que a H2F-hor e H2F-ite obtiveram valores próximos da formulação $\mathcal{TI}$ nas instâncias com $|I| = 20$ e $|I| = 40$. Porém, para as instâncias com $|I| = 80$, as duas versões da H2F apresentam valores superiores à formulação $\mathcal{TI}$. Esses valores estão associados aos valores negativos do indicador RDP na Tabela 4.3, que indica que, para essas instâncias, a H2F

encontrou melhores soluções que a formulação $\mathcal{TI}$. Por fim, embora a H2F-per não tenha apresentado resultados tão satisfatórios quanto as duas outras versões da H2F, nota-se que os valores médios percentuais de manutenções programadas apresentam relativamente proximidade aos valores dos demais métodos.

A Tabela 4.5 apresenta os resultados de tempos de execução dos métodos de solução e porcentagem de manutenções programadas para diferentes valores de $|M|$ e Φ. Ela está dividida em duas partes. A primeira parte apresenta resultados de tempo e a segunda parte mostra os resultados de percentual de manutenções programadas. Cada linha da tabela mostra o valor médio do tempo de execução ou percentual para um conjunto de instâncias que possuem a configuração apresentada nas duas primeiras linhas da tabela. As colunas $|M|$ e Φ indicam, respectivamente, o número de máquinas e a percentagem do tempo que separa as duas janelas das equipes

Tabela 4.4: Valores médios de tempo de execução e porcentagem de manutenções programadas de todos os métodos de solução para cada valor de $|I|$ e Ω.

		Form. $\mathcal{TI}$	**H2F-per**				**H2F-hor**				**H2F-ite**			
$\|I\|$	Ω		$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$	$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$	$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$
		tempo(s)												
20	**25%**	20,30	0,55	1,80	3,40	3,93	2,51	2,53	2,72	2,70	3,75	13,43	13,28	13,25
	50%	52,20	0,63	2,03	4,06	4,69	2,81	2,78	2,90	2,86	5,52	13,46	14,52	14,50
	80%	82,17	0,73	2,17	4,35	5,05	2,30	2,36	2,30	2,13	7,25	12,43	14,60	14,06
40	**25%**	389,67	1,11	3,86	6,98	8,31	6,88	6,65	6,77	6,63	8,15	28,34	30,83	28,61
	50%	795,90	1,35	4,55	9,57	10,61	7,92	8,55	8,03	7,80	10,04	26,77	30,36	26,31
	80%	1173,67	2,86	4,77	10,91	11,65	13,12	12,49	11,78	12,72	22,61	25,04	23,27	28,56
80	**25%**	1312,31	3,45	11,62	18,20	21,39	28,23	28,29	27,20	26,72	35,82	66,50	64,47	65,98
	50%	1683,39	3,45	11,66	22,90	28,15	37,55	36,38	40,76	40,55	50,36	71,44	68,64	72,50
	80%	1747,13	8,10	13,99	28,65	34,65	45,32	46,14	43,69	38,86	60,90	64,90	68,37	63,04
valor médio total		806,30	2,47	6,27	12,11	14,27	16,29	16,24	16,24	15,66	22,71	35,81	36,48	36,31
		% manutenções programadas												
20	**25%**	88,54	85,42	85,10	85,63	87,60	88,23	87,71	87,60	87,60	88,33	87,92	87,92	87,92
	50%	86,67	80,94	82,81	82,60	83,44	85,73	85,31	85,10	84,90	86,56	86,15	86,04	86,15
	80%	80,83	75,94	76,77	76,25	77,29	79,48	79,27	79,17	78,54	80,42	80,52	80,10	80,10
40	**25%**	87,92	84,53	85,36	85,94	85,99	87,86	87,50	87,60	87,45	87,86	87,86	87,92	87,81
	50%	82,14	77,81	78,18	79,64	79,58	81,82	81,15	81,56	81,25	82,08	81,93	82,03	81,88
	80%	68,70	64,43	64,48	65,99	64,74	67,81	66,77	67,40	66,61	68,07	67,14	67,92	67,24
80	**25%**	89,09	86,61	87,37	87,32	87,66	89,66	89,90	89,84	89,69	89,77	89,97	90,03	89,77
	50%	86,09	83,20	84,09	84,09	85,13	86,98	86,67	86,22	86,35	87,06	86,82	86,33	86,64
	80%	74,06	78,36	79,22	79,27	80,00	81,38	80,16	78,98	80,18	81,48	80,31	79,19	80,55
valor médio total		82,67	79,69	80,38	80,75	81,27	83,22	82,71	82,61	82,51	83,52	83,18	83,05	83,12

Tabela 4.5: Valores médios de tempo de execução e porcentagem de manutenções programadas de todos os métodos de solução para cada valor de $|M|$ e Φ.

		Form. $\mathcal{TI}$	**H2F-per**				**H2F-hor**				**H2F-ite**			
$\|M\|$	Φ		$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$	$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$	$\pi = 24$	$\pi = 48$	$\pi = 72$	$\pi = 96$
		Tempo de execução(s)												
2	**5%**	504,48	1,61	5,26	10,38	13,86	12,57	13,41	14,83	12,90	17,16	35,11	36,31	34,28
	15%	280,94	1,39	4,63	8,15	13,30	10,15	10,32	9,37	9,27	20,37	31,07	32,53	32,05
4	**5%**	1514,30	3,81	7,98	15,90	15,90	22,50	22,61	21,01	22,67	25,06	36,05	35,88	35,77
	15%	925,50	3,07	7,21	14,03	14,03	19,95	18,63	19,75	17,82	28,27	41,02	41,22	43,14
		% de manutenções programadas												
2	5%	88,08	85,20	85,57	85,13	85,82	87,74	87,58	87,70	87,59	87,91	87,89	87,88	87,87
	15%	82,16	78,60	79,39	78,73	79,19	81,64	81,38	81,13	81,26	82,04	81,86	81,91	81,97
4	5%	82,69	80,10	81,22	82,89	83,36	84,36	83,94	83,63	83,40	84,71	84,28	83,80	83,83
	15%	77,75	74,87	75,34	76,24	76,71	79,12	77,96	77,97	77,78	79,41	78,68	78,62	78,80

Os resultados da primeira parte da Tabela 4.5 mostram valores médios de tempo computacional menores nas instâncias com $|M| = 2$ que aquelas com $|M| = 4$. Além disso, nota-se também que as instâncias com $\Phi = 5\%$ tiveram valores médios maiores que as com $\Phi = 15\%$. Com relação à segunda parte, verifica-se que as instâncias com menor valor de Φ tiveram um percentual maior de manutenções programadas. Isso ocorre porque, com menor valor de Φ, tem-se um menor intervalo de tempo entre as janelas, logo tendo mais tempo disponível para programar manutenções.

4.4.2 Experimentos do conjunto de instâncias $\mathcal{M}$

A seguir é detalhado o procedimento de geração das instâncias do conjunto $\mathcal{M}$ e as características dessas instâncias. Em seguida, é apresentado e analisado os resultados de um experimento computacional com as respectivas instâncias.

4.4.2.1 Geração de instâncias para conjunto $\mathcal{M}$

As instâncias do conjunto $\mathcal{M}$ também foram geradas utilizando as instâncias apresentadas em Aquino et al. (2018a). Para a geração desse conjunto, foram consideradas as instâncias de Aquino et al. (2018a) com 150 e 300 manutenções. As instâncias de Aquino et al. (2018a) utilizadas para a geração dos conjuntos $\mathcal{P}$ e $\mathcal{M}$ têm uma diferença com relação à proporção do número de manutenções, máquinas e equipes. As instâncias do conjunto $\mathcal{P}$ têm, por exemplo, a configuração dada por 40 manutenções, 4 máquinas e 4 equipes. Já as instâncias do conjunto $\mathcal{M}$ têm, por exemplo, a configuração dada por 150 manutenções, 120 máquinas e 148 equipes. Com essa diferença, a geração de novas instâncias sofre uma modificação.

A principal modificação é na geração de manutenções com multi-habilidades. No conjunto $\mathcal{M}$, o número total de manutenções com multi-habilidade é dado por $\sum_{m}^{|M|} \Omega \times |\mathcal{I}_m|$, em que o subconjunto $\mathcal{I}_m \subseteq I$ representa as manutenções associadas às máquinas m. Nesse conjunto de instâncias, o parâmetro Ω indica a porcentagem de manutenções com multi-habilidade para cada subconjunto de manutenções associado a cada máquina. A soma de manutenções com multi-habilidade em todas as máquinas é o número total de cada instância. Para alterar as manutenções mono-habilidade para multi-habilidade é utilizado o mesmo procedimento do conjunto $\mathcal{P}$. No conjunto $\mathcal{M}$ foram utilizados os valores de Ω iguais a 25%, 50% e 80%. Para cada valor de Ω, foram geradas duas instâncias.

Para gerar instâncias com janelas nas equipes, foi utilizado o mesmo procedimento detalhado na Subseção 4.4.1.1 para o conjunto $\mathcal{P}$. No conjunto $\mathcal{M}$ foram utilizados os valores de Φ dados por 5% e 15%. Para cada valor de Φ, foram geradas duas instâncias.

Tabela 4.6: Características das instâncias do conjunto $\mathcal{M}$

$\|I\|$	$\|M\|$	$\|K\|$
150	120	148
150	93	92
150	101	71
300	239	221
300	179	158
300	181	75

Portanto, para cada instância de Aquino et al. (2018a), foram geradas 24 ($2 \times 2 \times 2 \times 3$) novas instâncias. As instâncias utilizadas do artigo citado são apresentadas na Tabela 4.6. Ao todo, foram adaptadas seis instâncias originais do estudo de Aquino et al. (2018a) e, com elas, foram geradas 144 instâncias (6×24). Todas essas instâncias estão disponíveis em `https://github.com/joaoluizandrade/LTPMOSP-MS/tree/main/medium`.

Esta seção compara os resultados da formulação $\mathcal{TI}$ com as três versões da H2F com diferentes valores de π na formulação da primeira fase. Para cada versão da H2F, foram realizados experimentos com os valores do parâmetro π dados por $\pi = 24$, 48, 96 e 192. Note que o maior valor desse parâmetro para o conjunto $\mathcal{M}$ é maior que o utilizado no conjunto $\mathcal{P}$. Isso ocorre porque o horizonte de planejamento das instâncias do conjunto $\mathcal{M}$ é mais longo. Como os parâmetros de entrada das manutenções e das equipes são fracionários, cada período das formulações $\mathcal{TI}$ e $\mathcal{F}_{\mathcal{II}}$ tem a duração de 0,1 hora.

Para os experimentos com a formulação $\mathcal{TI}$, o tempo limite de execução do *solver* CPLEX foi definido para 4 horas (ou 14.400 segundos), enquanto para as diferentes versões de H2F o tempo limite de execução é igual a 3600 segundos. Para se obter esse valor de tempo limite de execução nas heurísticas, os parâmetros de entrada associados com tempo de execução têm os valores iguais a $ti_1 = |I|$, $TL = 3600$ e $TL_2 = \frac{TL}{2}$. Note que o tempo do parâmetro ti_1 é igual, em segundos, ao número de manutenções.

A Tabela 4.7 apresenta e compara os resultados computacionais da formulação $\mathcal{TI}$ com as diferentes versões da H2F com quatro valores de π. Cada linha está associada a um método de solução e indica o valor médio ou percentual, considerando todas as instâncias do conjunto $\mathcal{M}$ que possuem o número de manutenções indicado na coluna $|I|$. A coluna *Método* indica o método de solução utilizado para solucionar as instâncias agrupadas. As colunas $gap_1(\%)$, $opt_1(\%)$, $gap_2(\%)$, $opt_2(\%)$, $t(s)$ e $t_{PH1}(s)$ são similares as da Tabela 4.3. A coluna $\%lb$ aparece apenas na Tabela 4.7 e indica a porcentagem de instâncias em que o limite inferior é válido. Ele é válido somente se a

Tabela 4.7: Resultados computacionais para a formulação $\mathcal{TI}$ e as diferentes versões da H2F para solucionar instâncias do conjunto $\mathcal{M}$.

$\|I\|$	**Método**	**gap$_1$(%)**	**%lb**	**%opt$_1$**	**gap$_2$(%)**	**%opt$_2$**	**t(s)**	**t$_{F1}$(s)**
150	Form. $\mathcal{TI}$	–	–	0,00	–	–	–	–
	H2F-per$_{24}$	41,38	70,83	30,56	31,17	30,56	118,63	49,79
	H2F-per$_{48}$	14,32	76,39	52,78	11,87	55,56	123,13	47,34
	H2F-per$_{96}$	6,11	98,61	58,33	5,91	61,11	121,81	25,68
	H2F-per$_{192}$	4,72	**100,00**	61,11	4,47	65,28	131,31	**8,65**
	H2F-hor$_{24}$	0,00	70,83	**70,83**	**1,55**	**70,83**	101,29	49,76
	H2F-hor$_{48}$	0,58	76,39	66,67	**1,55**	**70,83**	99,87	47,37
	H2F-hor$_{96}$	2,97	98,61	65,28	2,84	68,06	**79,88**	26,11
	H2F-hor$_{192}$	3,71	**100,00**	62,50	3,45	66,67	213,08	**8,73**
	H2F-ite$_{24}$	0,00	70,83	**70,83**	**1,55**	**70,83**	114,45	49,75
	H2F-ite$_{48}$	0,58	76,39	66,67	**1,55**	**70,83**	1264,09	47,54
	H2F-ite$_{96}$	1,81	98,61	66,67	1,61	**70,83**	824,61	25,90
	H2F-ite$_{192}$	3,38	**100,00**	62,50	3,13	66,67	1395,82	**8,74**
300	Form. $\mathcal{TI}$	–	–	–	–	–	–	–
	H2F-per$_{24}$	–	0,00	0,00	39,29	1,39	**453,88**	300,17
	H2F-per$_{48}$	22,83	30,56	0,00	38,36	2,78	468,35	264,57
	H2F-per$_{96}$	29,58	97,22	11,11	29,27	11,11	505,29	74,33
	H2F-per$_{192}$	37,90	**100,00**	9,72	33,75	9,72	960,88	**29,34**
	H2F-hor$_{24}$	–	0,00	0,00	14,13	16,67	1138,12	300,19
	H2F-hor$_{48}$	7,64	30,56	0,00	9,71	29,17	1591,69	264,91
	H2F-hor$_{96}$	17,56	97,22	23,61	18,43	23,61	1742,84	74,66
	H2F-hor$_{192}$	25,22	**100,00**	19,44	20,99	19,44	2523,15	**29,39**
	H2F-ite$_{24}$	–	0,00	0,00	12,97	16,67	1145,91	300,17
	H2F-ite$_{48}$	7,60	30,56	0,00	**6,18**	**30,56**	2777,91	265,31
	H2F-ite$_{96}$	11,32	97,22	**27,78**	11,31	27,78	2850,60	74,73
	H2F-ite$_{192}$	15,75	**100,00**	**27,78**	11,40	27,78	2874,74	**29,40**

O carácter "–" indica que nenhum resultado foi gerado para a respectiva coluna.

formulação $\mathcal{F}_I$ encontrar uma solução ótima dentro do limite de tempo de execução ti_1 e apenas na primeira ou única iteração do H2F. Vale destacar que o gap_1 é calculado somente nas instâncias com limite inferior válido. Logo, o valor médio da coluna $gap_1(\%)$ é calculado considerando apenas as instâncias que possuem um valor de gap_1. Outro ponto que deve ser destacado é que a porcentagem da coluna $opt_1(\%)$ considera todas as instâncias com o número de manutenções indicado na coluna $|I|$. Assim, o percentual da coluna $\%opt_1$ deve ser igual ou menor à coluna $\%lb$.

Nos experimentos com as instâncias do conjunto $\mathcal{M}$, a formulação $\mathcal{TI}$ não conseguiu gerar nenhuma solução pois a execução encontrou o limite do valor de memória do computador utilizado. Como não foi gerado nenhum resultado, foi adicionado o caractere "−−"nas colunas associadas à formulação $\mathcal{TI}$. Já a H2F, nas três versões

e com diferentes valores de π, conseguiu gerar soluções para todas as instâncias do conjunto $\mathcal{M}$, sendo algumas delas soluções exatas. Vale destacar que a formulação da segunda fase da H2F e a $\mathcal{TI}$ possuem semelhanças, conforme descrito na Subseção 4.3.2. Porém, a formulação $\mathcal{F}_{\mathcal{II}}$ conseguiu gerar soluções para o problema em estudo. Isso ocorre porque essa formulação tem dimensões menores, devido à solução da primeira fase.

As diferentes porcentagens da coluna $\%lb$ e tempos da coluna $t_{F_I}(s)$ para cada valor π mostram a influência do parâmetro de entrada nos resultados. A formulação $\mathcal{F}_I$ com $\pi = 192$ gerou os melhores resultados e encontrou solução ótima para a fase 1 dentro do limite de tempo em 100% das instâncias. Além disso, obteve o menor valor médio de tempo, enquanto a formulação da primeira fase com $\pi = 24$ não conseguiu gerar nenhum limite inferior válido para gap_1 nas instâncias com $|I| = 300$. Note que o valor médio de tempo da coluna $t_{F_I}(s)$ está próximo do tempo de ti_1. Além disso, como não há nenhum limite inferior válido, não é possível calcular gap_1 para nenhuma instância.

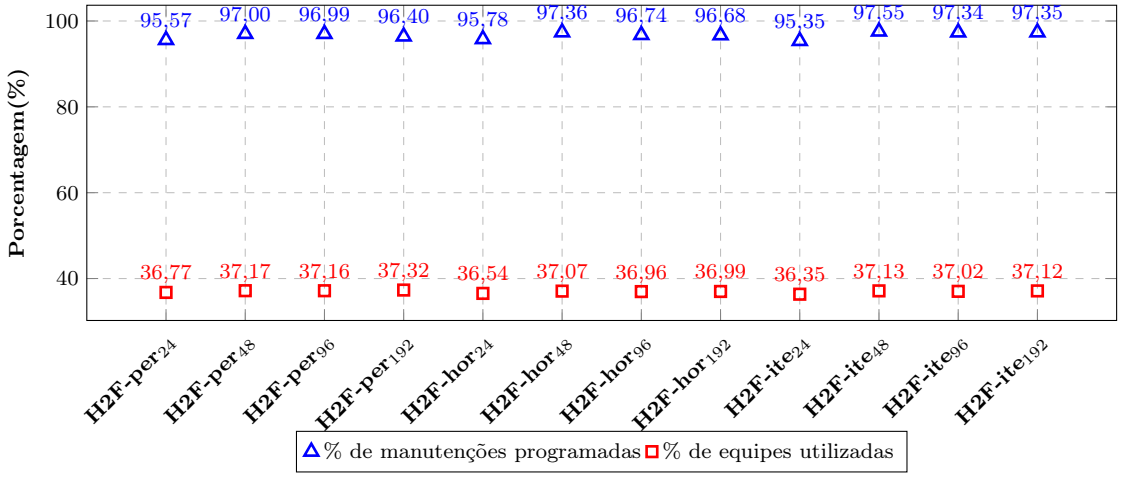
Ao analisar os valores da coluna gap_1, nota-se que, para os métodos H2F-hor$_{24}$ e H2F-iter$_{24}$, o valor médio é igual a zero. Isso indica que os métodos conseguiram encontrar solução ótima em todas as instâncias que obtiveram um limite inferior válido. Isso pode ser visto nos valores percentuais iguais das colunas $\%lb$ e $\%opt_1$.

Considerando todos os limites inferiores gerados pelos diferentes valores de π, é possível calcular o gap_2 para todas as instâncias do conjunto $\mathcal{M}$ e de todos os métodos. Ao analisar os valores da coluna gap_2 para as instâncias com $|I| = 150$, verifica-se que H2F-hor e H2F-ite obtiveram valores médios inferiores a 4%. Vale destacar que, para H2F-hor$_{24}$, H2F-hor$_{48}$, H2F-iter$_{24}$ e H2F-iter$_{48}$, o valor médio foi inferior a 2%, indicando soluções próximas da ótima. Esses baixos valores de gap_2 estão associados aos bons limites inferiores gerados pela formulação $\mathcal{F}_I$. Além disso, esses métodos conseguiram determinar a solução ótima em aproximadamente 71% das instâncias com $|I| = 150$. Note que a H2F-hor$_{48}$ conseguiu obter esses bons resultados em aproximadamente 100 segundos.

Quando analisados os valores da coluna gap_2 para instâncias com $|I| = 300$, observa-se um aumento com relação aos valores médios para as instâncias com $|I| = 150$. No entanto, o H2F-ite consegue ter valores médios próximos de 12% nas versões da heurística com $\pi = 24$, 96 e 192; para a versão com $\pi = 48$, o valor médio é próximo de 6%. Além disso, a H2F-iter$_{48}$ conseguiu encontrar solução ótima em aproximadamente 30% das instâncias com $|I| = 300$. Quando consideradas todas as instâncias do conjunto $\mathcal{M}$, a H2F-iter$_{48}$ encontrou solução ótima em mais de 50% das instâncias.

A Figura 4.6 mostra o valor médio percentual de manutenções programadas e

Figura 4.6: Valores médios de porcentagem de manutenções programadas e equipes utilizadas no planejamento para as diferentes versões da H2F.



equipes utilizadas para as diferentes versões da H2F. Observa-se que, embora as versões da H2F gerem diferentes qualidades de soluções, o percentual de manutenções programadas e equipes utilizadas nas soluções permanece relativamente próximo.

4.5 Conclusões parciais para o estudo do LTPMOSP-MS

Este capítulo abordou um método de solução para o LTPMOSP-MS. Para solucionar este problema de natureza $\mathcal{NP}$-difícil, foram propostas três versões de uma nova heurística de duas fases, denominada H2F. A primeira fase consiste em atribuir as manutenções às equipes e períodos, enquanto a segunda fase sequencia as manutenções nas equipes, respeitando todas as restrições do problema. A alocação da primeira fase é definida por uma formulação matemática baseada em variáveis indexadas no tempo. A estratégia de divisão utilizada nessa formulação é nova na literatura e é baseada na formulação da Seção 3.5. A primeira fase da H2F é responsável por transformar o LTPMOSP-MS em um ou vários subproblemas de menor dimensão, solucionados na segunda fase. Na H2F-per, o horizonte é dividido em subproblemas de menor dimensão e são solucionados independentemente e em sequência. Na H2F-hor, é solucionado um único subproblema para todo o horizonte. A H2F-ite utiliza a mesma estratégia da H2F-hor, porém, após solucionar o subproblema, pode ser gerado um pseudo-corte. Ele é adicionado na formulação da primeira fase e ambas as fases são executadas novamente.

A formulação $\mathcal{TI}$, da Seção 4.2, e as diferentes versões da H2F, foram testadas com instâncias que variam entre 20 e 300 manutenções. Para um conjunto de instân-

cias com 20 a 80 manutenções, a formulação $\mathcal{TI}$ conseguiu gerar soluções exatas para aproximadamente 82% das instâncias. A H2F-ite com $\pi = 24$ conseguiu encontrar soluções ótimas em aproximadamente 79% das instâncias, porém gerou limites inferiores e superiores mais fortes em um tempo significativamente menor. Vale destacar que todas as versões da H2F conseguiram encontrar limites superiores melhores do que da formulação $\mathcal{TI}$ para as instâncias com 80 manutenções. Para o conjunto de instâncias com 150 e 300 manutenções, a formulação proposta não conseguiu gerar nenhuma solução, enquanto que a H2F conseguiu solucionar todas as instâncias. A H2F-hor e H2F-ite encontraram soluções ótimas para aproximadamente 71% das instâncias com 150 manutenções em um tempo computacional médio de 115 segundos. Para as instâncias com 300 manutenções, a H2F-iterativa com $\pi = 48$ encontrou solução ótima em aproximadamente 31% das instâncias. No total, a H2F-ite com $\pi = 48$ encontrou solução ótima em aproximadamente 50% das instâncias com 150 e 300 manutenções.

Capítulo 5

Considerações finais e trabalhos futuros

As considerações finais e os trabalhos futuros são apresentados neste capítulo. A Seção 5.1 apresenta as considerações finais da pesquisa. A Seção 5.2 relaciona os trabalhos publicados durante o doutoramento. Por fim, a Seção 5.3 discute possíveis linhas de pesquisa para continuar este estudo.

5.1 Considerações finais

Nesta tese, foram pesquisados dois importantes e desafiadores problemas de otimização. O primeiro deles foi o Problema de Planejamento de Ordens de Manutenção Preventiva de Longo Prazo (LTPMOSP). Ele foi introduzido em Aquino et al. (2018a) e busca determinar quais manutenções serão programadas, dado um conjunto delas, em que instante elas serão executadas e por qual equipe. Cada manutenção tem uma janela específica para sua execução, em que ela pode ser executada. Além disso, no LTPMOSP, cada manutenção requer uma única habilidade para ser executada e cada equipe possui uma única habilidade. Assim, cada manutenção pode ser executada apenas por um conjunto particular de equipes. O segundo problema é o Problema de Planejamento de Manutenções Preventivas com Multi-habilidades (LTPMOSP-MS). Ele é introduzido nesta tese e é uma extensão do LTPMOSP. Nele, as manutenções podem requerer uma ou mais habilidades para serem executadas. Para as manutenções com multi-habilidade serem executadas, é necessário atribuí-las a um número de equipes igual às habilidades que elas requerem. Outra diferença entre o LTPMOSP-MS e o LTPMOSP é com relação à disponibilidade das equipes. Em LTPMOSP cada equipe tem uma única janela de disponibilidade para executar manutenções, porém, no LTPMOSP-MS, as equipes possuem duas janelas com um intervalo de indisponi-

bilidade entre elas. Ambos os problemas têm como objetivo minimizar o número de equipes utilizadas no planejamento e a soma das penalidades por manutenções não programadas. A revisão de literatura realizada nesta tese destacou que as características dos dois problemas podem ser encontradas em outros estudos da literatura, porém, até onde sabemos, não todas elas.

Devida à natureza $\mathcal{NP}$-difícil do LTPMOSP e à larga escala de uma instância real do problema, as pesquisas anteriores a esta deram grande foco para métodos heurísticos para a sua resolução. Este trabalho seguiu em outra linha de pesquisa, dando uma ênfase maior em métodos exatos. Foram propostas duas novas formulações matemáticas. A primeira formulação é baseada em variáveis indexadas no tempo e a segunda combina variáveis indexadas no tempo e de precedência. Até onde sabemos, a segunda propõe uma nova estratégia na literatura. No entanto, para lidar com instâncias de grande escala e real, foi desenvolvido um algoritmo baseado na meta-heurística GRASP. Por fim, foi proposto um método de solução híbrido exato. Os resultados dos experimentos computacionais mostram que as formulações matemáticas propostas são superiores, em relação a encontrar soluções ótimas e limites fortes, do que a formulação proposta por Aquino et al. (2018a). Com destaque para a segunda formulação, que conseguiu encontrar solução ótima em instâncias com 150, 300 e 600 manutenções. Além disso, o algoritmo GRASP desenvolvido nesta pesquisa mostrou-se ser mais eficiente que os algoritmos propostos em trabalhos anteriores, encontrando limites superiores melhores em tempo de execução menor.

Para solucionar o LTPMOSP-MS, foi proposta uma formulação matemática baseada em variáveis indexadas no tempo e uma nova heurística de duas fases. Nas três versões da heurística, a primeira fase atribui as manutenções às equipes e a segunda fase sequencia as manutenções nas equipes. Embora a heurística não seja um algoritmo exato, ela é capaz de encontrar soluções ótimas. Os resultados dos experimentos computacionais mostram um desempenho satisfatório da formulação matemática para instâncias de pequena escala, encontrando limites fortes e soluções ótimas para a maioria dessas instâncias. Porém, para as instâncias de média escala, a formulação não conseguiu gerar nenhuma solução. Com relação aos resultados da heurística, duas versões dela conseguiram encontrar um número de soluções ótimas muito próximo da formulação para as instâncias de pequena escala. Porém, elas foram capazes de gerar limites mais fortes para essas instâncias. Para as instâncias de média escala, a versão iterativa da heurística conseguiu gerar solução ótima para 50% das instâncias. Com esses resultados, a versão mais indicada da heurística de duas fases é a iterativa, uma vez que gerou os melhores resultados para todas as instâncias.

5.2 Trabalhos derivados desta pesquisa

Os trabalhos a seguir resultam dessa tese:

(i) DE ANDRADE, J.L.M., SOUZA, M.J.F., DE SÁ, E.M., MENEZES, G.C. e de Souza, S.R.. Um algoritmo GRASP eficiente para um problema de planejamento de ordens de manutenção de longo prazo. In: **XLI Congresso Nacional de Matemática Aplicada e Computacional**, 2022, Campinas. Um algoritmo GRASP eficiente para um problema de planejamento de ordens de manutenção de longo prazo, 2022. v. 9. *https://doi.org/10.5540/03.2022.009.01.0314*

(ii) DE ANDRADE, J.L.M., SOUZA, M.J.F., DE SÁ, E.M., MENEZES, G.C. e de Souza, S.R. Uma nova formulação matemática para um problema de planejamento de ordens de manutenção preventiva de longo prazo. In: **LV Simpósio Brasileiro de Pesquisa Operacional (SBPO 2023)**, 2023, São José dos Campos. Anais do Simpósio Brasileiro de Pesquisa Operacional, 2023. *https://doi.org/10.59254/sbpo-2023-174884*

(iii) DE ANDRADE, J.L.M., SOUZA, M.J.F., DE SÁ, E.M., MENEZES, G.C. e de Souza, S.R. (2024). *Formulations and heuristic for the long-term preventive maintenance order scheduling problem.* **Computers & Operations Research**, 170, p.106781. *https://doi.org/10.1016/j.cor.2024.106781.*

(iv) DE ANDRADE, J.L.M., SOUZA, M.J.F., DE SÁ, E.M., MENEZES, G.C. e de Souza, S.R. Uma formulação matemática para o problema de planejamento de manutenções preventivas com multi-habilidades In: **LVI Simpósio Brasileiro de Pesquisa Operacional (SBPO 2024)**, 2024, Fortaleza. Anais do Simpósio Brasileiro de Pesquisa Operacional, 2024. *https://doi.org/10.59254/sbpo-2024-193590*

(v) DE ANDRADE, J.L.M., SOUZA, M.J.F., DE SÁ, E.M., MENEZES, G.C. and de Souza, S.R. (2025). *Matheuristics for a preventive maintenance scheduling problem with multi-skilled.* **Submetido a uma revista científica.**

5.3 Trabalhos futuros

Apesar dos resultados muito importantes, ainda existem desafios e oportunidades de pesquisa para ambos problemas abordados nesta tese. Nos experimentos computacionais de ambos os problemas, as formulações baseadas em variáveis conseguiram gerar soluções apenas para as instâncias de pequena escala. Para as instâncias com

escala maior a essa, as dimensões da formulação ficam muito grandes e requerem grande quantidade de memória. Na literatura já existem trabalhos, como de Akker et al. (2000) e Bigras et al. (2008), que contornam esse problema utilizando o método de geração de coluna. Portanto, uma alternativa de pesquisa futura é adaptar as formulações propostas para possibilitar a implementação desse método.

Outra alternativa de pesquisa futura para os dois problemas é a consideração de incerteza. Embora Coelho et al. (2026) já tenha abordado incerteza no tempo de execução das manutenções, ainda há outras formas na literatura abordar incerteza. Uma delas é a abordagem robusta Bertsimas & Sim (2003). Além disso pode ser considerado incerteza em outro parâmetros como o tempo de disponibilidade de uma equipe. Uma vez que pode ocorrer incapacidade inesperada de uma equipe.

Os resultados dos experimentos computacionais do método de solução proposto para o LTPMOSP-MS mostram que a formulação da primeira fase foi capaz de gerar limites inferiores que convergem com os limites superiores gerados no final da segunda fase. No entanto, isso não foi possível para todas as instâncias. Logo, uma sugestão para trabalhos futuros é o desenvolvimento de melhorias na formulação. Outra sugestão é o desenvolvimento de uma heurística ou meta-heurística para a segunda fase da H2F. A diferença dos tempos das colunas $t(s)$ e t_{F1} mostra que a segunda fase possui a maior parte do tempo de execução da H2F. Além disso, a formulação da segunda fase é baseada em variáveis que requerem grande quantidade de memória. Assim, uma heurística pode tanto reduzir o tempo computacional, bem como lidar com soluções de instâncias de maior escala. Por fim, mais uma sugestão seria desenvolver uma versão H2F iterativa utilizando a estratégia da segunda fase da H2F-per. Uma vantagem dessa versão da H2F são os subproblemas na segunda fase. O desenvolvimento de pseudo-cortes que melhorem a atribuição das manutenções a períodos pode deixar o método de solução mais poderoso.

Referências

Afshar-Nadjafi, B. (2021). Multi-skilling in scheduling problems: A review on models, methods and applications. *Computers & Industrial Engineering*, *151*, 107004. doi: doi.org/10.1016/j.cie.2020.107004.

Agnetis, A., Billaut, J.-C., Pinedo, M., & Shabtay, D. (2025). Fifty years of research in scheduling — theory and applications. *European Journal of Operational Research*, *327*, 367–393. doi: https://doi.org/10.1016/j.ejor.2025.01.034.

Akker, J., Hurkens, C., & Savelsbergh, M. (2000). Time-indexed formulations for machine scheduling problems: Column generation. *INFORMS Journal on Computing*, *12*, 111–124. doi: doi.org/10.1287/ijoc.12.2.111.11896.

Almeida, B. F., Correia, I., & Saldanha-da Gama, F. (2018). A biased random-key genetic algorithm for the project scheduling problem with flexible resources. *TOP*, *26*, 283–308. doi: doi.org/10.1007/s11750-018-0472-9.

Almeida, B. F., Correia, I., & Saldanha-da Gama, F. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. *International Transactions in Operational Research*, *26*, 946–967. doi: doi.org/10.1111/itor.12568.

Anoshkina, Y., & Meisel, F. (2020). Interday routing and scheduling of multi-skilled teams with consistency consideration and intraday rescheduling. *EURO Journal on Transportation and Logistics*, *9*, 100012. `https://doi.org/10.1016/j.ejtl.2020.100012`.

Aquino, R. D., Chagas, J. B. C., & Souza, M. J. F. (2018a). A mixed-integer linear programming model and a simulated annealing algorithm for the long-term preventive maintenance scheduling problem. In A. Abraham, P. K. Muhuri, A. K. Muda, & N. Gandhi (Eds.), *Intelligent Systems Design and Applications (ISDA 2017)* (pp. 144–153). Cham: Springer International Publishing volume 736 of *Advances in Intelligent Systems and Computing*. doi: doi.org/10.1007/978-3-319-76348-4_15.

Aquino, R. D., Chagas, J. B. C., & Souza, M. J. F. (2018b). A variable neighborhood search algorithm for the long-term preventive maintenance scheduling problem. In *Proceedings of the 20th International Conference on Enterprise Information Systems - Volume 1: ICEIS* (pp. 303–310). INSTICC Funchal, Madeira, Portugal: SciTePress. URL: https://www.scitepress.org/papers/2018/66897/66897.pdf.

Aquino, R. D., Chagas, J. B. C., & Souza, M. J. F. (2019). Abordagem exata e heurística para o problema de planejamento de ordens de manutenção de longo prazo: um estudo de caso industrial de larga escala. *Pesquisa Operacional para o Desenvolvimento*, *11*, 159–182. URL: https://doi.org/10.4322/PODes.2019.012.

Balas, E. (1985). On the facial structure of scheduling polyhedra. In R. W. Cottle (Ed.), *Mathematical Programming Essays in Honor of George B. Dantzig Part I* (pp. 179–218). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/BFb0121051. doi: 10.1007/BFb0121051.

Barghi, B., & Sikari, S. S. (2022). Meta-heuristic solution with considering setup time for multi-skilled project scheduling problem. *Operations Research Forum*, *3*, 16. URL: https://doi.org/10.1007/s43069-021-00117-5.

Barros, V. H., Costa, T. S., Oliveira, A. C., & Lorena, L. A. (2011). Model and heuristic for berth allocation in tidal bulk ports with stock level constraints. *Computers & Industrial Engineering*, *60*, 606–613. https://doi.org/10.1016/j.cie.2010.12.018.

Berghman, Lotte, Spieksma, Frits C.R., & T´kindt, Vincent (2021). Solving a time-indexed formulation for an unrelated parallel machine scheduling problem by preprocessing and cutting planes. *RAIRO-Oper. Res.*, *55*, S1747–S1765. https://doi.org/10.1051/ro/2020031.

Bertsimas, D., & Sim, M. (2003). Robust discrete optimization and network flows. *Mathematical Programming*, *98*, 49–71. URL: https://doi.org/10.1007/s10107-003-0396-4. doi: 10.1007/s10107-003-0396-4.

Bigras, L.-P., Gamache, M., & Savard, G. (2008). Time-indexed formulations and the total weighted tardiness problem. *INFORMS Journal on Computing*, *20*, 133–142. https://doi.org/10.1287/ijoc.1070.0225.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. *European Journal of Operational Research*, *51*, 283–300. doi: https://doi.org/10.1016/0377-2217(91)90304-E.

Bocewicz, G., Golinska-Dawson, P., Szwarc, E., & Banaszak, Z. (2023). Preventive maintenance scheduling of a multi-skilled human resource-constrained project's portfolio. *Engineering Applications of Artificial Intelligence*, *119*, 105725. https://doi.org/10.1016/j.engappai.2022.105725.

Brucker, P. (2004). *Scheduling Algorithms*. Berlin: Springer International Publishing. https://link.springer.com/book/10.1007/978-3-540-69516-5.

Brucker, P., Dhaenens-Flipo, C., Knust, S., Kravchenko, S. A., & Werner, F. (2002). Complexity results for parallel machine problems with a single server. *Journal of Scheduling*, *5*, 429–457. https://doi.org/10.1002/jos.120.

Brucker, P., & Knust, S. (2009). Complexity results for scheduling problems. URL: http://www2.informatik.uni-osnabrueck.de/knust/class/.

Cakici, E., & Mason, S. J. (2007). Parallel machine scheduling subject to auxiliary resource constraints. *Production Planning & Control*, *18*, 217 – 225.

Carrabs, F., Cerulli, R., Mansini, R., Moreschini, L., & Serra, D. (2024). Solving the set covering problem with conflicts on sets: A new parallel grasp. *Computers & Operations Research*, (p. 106620). URL: https://www.sciencedirect.com/science/article/pii/S0305054824000923. doi: https://doi.org/10.1016/j.cor.2024.106620.

Cesaret, B., Oğuz, C., & Sibel Salman, F. (2012). A tabu search algorithm for order acceptance and scheduling. *Computers & Operations Research*, *39*, 1197–1205. https://doi.org/10.1016/j.cor.2010.09.018.

Chansombat, S., Pongcharoen, P., & Hicks, C. (2019). A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry. *International Journal of Production Research*, *57*, 61–82. https://doi.org/10.1080/00207543.2018.1459923.

Chaurasia, S. N., & Singh, A. (2017). Hybrid evolutionary approaches for the single machine order acceptance and scheduling problem. *Applied Soft Computing*, *52*, 725–747. https://doi.org/10.1016/j.asoc.2016.09.051.

Cheimanoff, N., Féniès, P., Kitri, M. N., & Tchernev, N. (2023). Exact and metaheuristic approaches to solve the integrated production scheduling, berth allocation and storage yard allocation problem. *Computers & Operations Research*, *153*, 106174. URL: https://www.sciencedirect.com/science/article/pii/S0305054823000382. doi: https://doi.org/10.1016/j.cor.2023.106174.

Chen, C., Rong; Liang, & Gu, D. (2017). It project portfolio scheduling and multi-skilled staff assignment with ant colony optimization algorithm. In *WHICEB 2016 Proceedings*. volume 9. https://aisel.aisnet.org/whiceb2016/9/.

Chen, R., Liang, C., Gu, D., & Zhao, H. (2020). A competence-time-quality scheduling model of multi-skilled staff for it project portfolio. *Computers & Industrial Engineering*, *139*, 106183. https://doi.org/10.1016/j.cie.2019.106183.

Chudak, F. A., & Hochbaum, D. S. (1999). A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, *25*, 199–204. doi: https://doi.org/10.1016/S0167-6377(99)00056-5.

Coelho, D. G., Souza, M. J. F., & Cota, L. P. (2026). A simheuristic-based algorithm for the stochastic long-term maintenance scheduling problem. *International Transactions in Operational Research*, *33*, 268–296. URL: https://doi.org/10.1111/itor.70021.

Dauzere-Peres, S. (1997). An efficient formulation for minimizing the number of late jobs in single-machine scheduling. In *1997 IEEE 6th International Conference on Emerging Technologies and Factory Automation Proceedings, EFTA '97* (pp. 442–445). doi: 10.1109/ETFA.1997.616311.

Dauzère-Pérès, S., & Sevaux, M. (2003). Using lagrangean relaxation to minimize the weighted number of late jobs on a single machine. *Naval Research Logistics (NRL)*, *50*, 273 – 288. doi: 10.1002/nav.10056.

de Weerdt, M., Baart, R., & He, L. (2021). Single-machine scheduling with release times, deadlines, setup times, and rejection. *European Journal of Operational Research*, *291*, 629–639. https://doi.org/10.1016/j.ejor.2020.09.042.

Dyer, M. E., & Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, *26*, 255–270. doi: https://doi.org/10.1016/0166-218X(90)90104-K.

Emami, S., Moslehi, G., & Sabbagh, M. (2017). A benders decomposition approach for order acceptance and scheduling problem: a robust optimization approach. *Computational and Applied Mathematics*, *36*, 1471–1515. https://doi.org/10.1007/s40314-015-0302-8.

Emami, S., Sabbagh, M., & Moslehi, G. (2016). A lagrangian relaxation algorithm for order acceptance and scheduling problem: a globalised robust optimisation approach. *International Journal of Computer Integrated Manufacturing*, *29*, 535–560. https://doi.org/10.1080/0951192X.2015.1068452.

Feo, T. A., & Resende, M. G. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, *8*, 67–71. https://doi.org/10.1016/0167-6377(89)90002-3.

Feo, T. A., & Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*, 109–133. https://doi.org/10.1007/BF01096763.

Fox, K., Gavish, B., & Graves, S. (1980). An n-constraint formulation of the (time-dependent) traveling salesman problem. *Operations Research*, *28*, 1018–1021. URL: http://doi.org/10.1287/opre.28.4.1018.

Froger, A., Gendreau, M., Mendoza, J. E., Éric Pinson, & Rousseau, L.-M. (2016). Maintenance scheduling in the electricity industry: A literature review. *European Journal of Operational Research*, *251*, 695–706. https://doi.org/10.1016/j.ejor.2015.08.045.

Geramipour, S., Moslehi, G., & Reisi-Nafchi, M. (2017). Maximizing the profit in customer's order acceptance and scheduling problem with weighted tardiness penalty. *Journal of the Operational Research Society*, *68*, 89–101. https://doi.org/10.1057/s41274-016-0007-4.

Geurtsen, M., Didden, J. B., Adan, J., Atan, Z., & Adan, I. (2022). Production, maintenance and resource scheduling: A review. *European Journal of Operational Research*, . https://doi.org/10.1016/j.ejor.2022.03.045.

Gholami, O., Sotskov, Y. N., Werner, F., & Zatsiupo, A. S. (2019). Heuristic algorithms to maximize revenue and the number of jobs processed on parallel machines. *Automation and Remote Control*, *80*, 297–316. https://doi.org/10.1134/S0005117919020085.

Glover, F., Lasdon, L., Plummer, J., Duarte, A., Marti, R., Laguna, M., & Rego, C. (2011). Pseudo-cut strategies for global optimization. *Int. J. of Applied Metaheuristic Computing*, *2*, 1–12. doi: 10.4018/jamc.2011100101.

Graham, R., Lawler, E., Lenstra, J., & Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In P. Hammer, E. Johnson, & B. Korte (Eds.), *Discrete Optimization II* (pp. 287–326). Elsevier volume 5 of *Annals of Discrete Mathematics*. https://doi.org/10.1016/S0167-5060(08)70356-X.

Hart, J., & Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters*, *6*, 107–114. doi: https://doi.org/10.1016/0167-6377(87)90021-6.

Henao, C. A., Muñoz, J. C., & Ferrer, J. C. (2015). The impact of multi-skilling on personnel scheduling in the service sector: a retail industry case. *Journal of the Operational Research Society*, *66*, 1949–1959. https://doi.org/10.1057/jors.2015.9.

Jonge, B. D., & Scarf, P. A. (2020). A review on maintenance optimization. *European Journal of Operational Research*, *285*, 805–824. https://doi.org/10.1016/j.ejor.2019.09.047.

Karam, A., Attia, E.-A., & Duquenne, P. (2017). A MILP model for an integrated project scheduling and multi-skilled workforce allocation with flexible working hours. *IFAC-PapersOnLine*, *50*, 13964–13969. https://doi.org/10.1016/j.ifacol.2017.08.2221.

Kasirzadeh, A., Saddoune, M., & Soumis, F. (2017). Airline crew scheduling: models, algorithms, and data sets. *EURO Journal on Transportation and Logistics*, *6*, 111–137. doi: https://doi.org/10.1007/s13676-015-0080-x.

Lasserre, J. B., & Queyranne, M. (1992). Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling. In *Conference on Integer Programming and Combinatorial Optimization*.

Li, Y., Chu, F., Zheng, F., & Liu, M. (2022). A bi-objective optimization for integrated berth allocation and quay crane assignment with preventive maintenance activities. *IEEE Transactions on Intelligent Transportation Systems*, *23*, 2938–2955. https://doi.org/10.1109/TITS.2020.3023701.

Liang, Y., Wu, W., Wang, K., & Hu, C. (2019). Online multi-skilled task assignment on road networks. *IEEE Access*, *7*, 57371–57382. doi: 10.1109/ACCESS.2018.2879331.

Lin, J., Zhu, L., & Gao, K. (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, *140*, 112915. https://doi.org/10.1016/j.eswa.2019.112915.

Lin, S.-W., & Ying, K.-C. (2013). Increasing the total net revenue for single machine order acceptance and scheduling problems using an artificial bee colony algorithm. *Journal of the Operational Research Society*, *64*, 293–311. https://doi.org/10.1057/jors.2012.47.

Liu, P., & Lu, X. (2020). New approximation algorithms for machine scheduling with rejection on single and parallel machine. *Journal of Combinatorial Optimization*, *40*, 929–952. https://doi.org/10.1007/s10878-020-00642-9.

Liu, Z., Liu, J., Zhuang, C., & Wan, F. (2022). Multi-objective complex product assembly scheduling problem considering parallel team and worker skills. *Journal of Manufacturing Systems*, *63*, 454–470. doi: https://doi.org/10.1016/j.jmsy.2022.05.003.

López–Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., & Stützle, T. (2016). The IRACE package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, *3*, 43–58. https://doi.org/10.1016/j.orp.2016.09.002.

Magnoga, B. A. K., & Cota, L. P. (2023). Algoritmos heurísticos para o sequenciamento de ordens de manutenção de tecnologia em um horizonte de curto prazo. In *Anais do LV Simpósio Brasileiro de Pesquisa Operacional (SBPO 2023)* (pp. 1–12). Belo Horizonte, MG, Brasil: SOBRAPO. doi: 10.59254/sbpo-2023-174802.

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*, 219–223.

Nadali, F., Iranpoor, M., & Malekian, Y. (2024). The integrated planning of outgoing coil selection for retrieval, multi-crane scheduling, and location assignment to the incoming and blocking coils in steel coil warehouses. *Computers & Industrial Engineering*, *190*, 110058. URL: https://www.sciencedirect.com/science/article/pii/S0360835224001797. doi: https://doi.org/10.1016/j.cie.2024.110058.

Naderi, B., & Roshanaei, V. (2020). Branch-relax-and-check: A tractable decomposition method for order acceptance and identical parallel machine scheduling. *European Journal of Operational Research*, *286*, 811–827. https://doi.org/10.1016/j.ejor.2019.10.014.

Nemhauser, G. L., & Savelsbergh, M. W. P. (1992). A cutting plane algorithm for the single machine scheduling problem with release times. In M. Akgül, H. W. Hamacher, & S. Tüfekçi (Eds.), *Combinatorial Optimization* (pp. 63–83). Berlin, Heidelberg: Springer Berlin Heidelberg.

Nesello, V., Subramanian, A., Battarra, M., & Laporte, G. (2018). Exact solution of the single-machine scheduling problem with periodic maintenances and sequence-dependent setup times. *European Journal of Operational Research*, *266*, 498–507. doi: https://doi.org/10.1016/j.ejor.2017.10.020.

Ngoo, C. M., Goh, S. L., Sze, S. N., Sabar, N. R., Hijazi, M. H. A., & Kendall, G. (2024). A survey of mat-heuristics for combinatorial optimisation problems: Variants, trends and opportunities. *Applied Soft Computing*, *164*, 111947. URL: https://www.sciencedirect.com/science/article/pii/S156849462400721X. doi: https://doi.org/10.1016/j.asoc.2024.111947.

Nobibon, F. T., & Leus, R. (2011). Exact algorithms for a generalization of the order acceptance and scheduling problem in a single-machine environment. *Computers & Operations Research*, *38*, 367–378. URL: https://doi.org/10.1016/j.cor.2010.06.003. Project Management and Scheduling.

Ou, J., & Zhong, X. (2017). Bicriteria order acceptance and scheduling with consideration of fill rate. *European Journal of Operational Research*, *262*, 904–907. https://doi.org/10.1016/j.ejor.2017.04.036.

Oğuz, C., Sibel Salman, F., & Bilgintürk Yalçın, Z. (2010). Order acceptance and scheduling decisions in make-to-order systems. *International Journal of Production Economics*, *125*, 200–211. https://doi.org/10.1016/j.ijpe.2010.02.002.

Pessan, C., Bellenguez-Morineau, O., & Neron, E. (2007). Multi-skill project scheduling problem and Total Productive Maintenance. In *3rd Multidisciplinary International Conference on Scheduling: Theory and Application* (pp. 608–610). Paris, France. https://hal.science/hal-01025570.

Pessoa, A., Uchoa, E., de Aragão, M. P., & Rodrigues, R. (2010). Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, *2*, 259–290. https://doi.org/10.1007/s12532-010-0019-z.

Pinedo, M. L. (2022). *Scheduling: Theory, Algorithms, and Systems*. (sixth ed.). Cham: Springer. https://doi.org/10.1007/978-3-031-05921-6.

Pisinger, D. (1999). An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research*, *114*, 528–541. URL: https://www.sciencedirect.com/science/article/pii/S0377221798001209. doi: https://doi.org/10.1016/S0377-2217(98)00120-9.

Potts, C. N. (1980). An algorithm for the single machine sequencing problem with precedence constraints. In V. J. Rayward-Smith (Ed.), *Combinatorial Optimization II* (pp. 78–87). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/BFb0120909. doi: 10.1007/BFb0120909.

Prais, M., & Ribeiro, C. (2000a). Parameter variation in grasp procedures. *Investigación Operativa*, *9*, 1–20.

Prais, M., & Ribeiro, C. C. (2000b). Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. *INFORMS Journal on Computing*, *12*, 164–176. doi: 10.1287/ijoc.12.3.164.12639.

Pérez, C., Climent, L., Nicoló, G., Arbelaez, A., & Salido, M. A. (2023). A hybrid metaheuristic with learning for a real supply chain scheduling problem. *Engineering Applications of Artificial Intelligence*, *126*, 107188. URL: https://www.sciencedirect.com/science/article/pii/S0952197623013726. doi: https://doi.org/10.1016/j.engappai.2023.107188.

Qin, S., Liu, S., & Kuang, H. (2016). Piecewise linear model for multiskilled workforce scheduling problems considering learning effect and project quality. *Mathematical Problems in Engineering*, *2016*, 3728934. https://doi.org/10.1155/2016/3728934.

Queyranne, M. (1993). Structure of a simple scheduling polyhedron. *Mathematical Programming*, *58*, 263–285. doi: 10.1007/BF01581271.

Queyranne, M., & Schulz, A. S. (1994). *Polyhedral approaches to machine scheduling*. Berlin: TU, Fachbereich 3. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3b7415b4fcb83f0000a3f4acfbb42cc1dcc489ec.

Queyranne, M., & Wang, Y. (1991). Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research*, *16*, 1–20. URL: http://www.jstor.org/stable/3689846.

Rebai, M., Kacem, I., & Adjallah, K. H. (2012). Earliness-tardiness minimization on a single machine to schedule preventive maintenance tasks: metaheuristic and exact methods. *Journal of Intelligent Manufacturing*, *23*, 1207–1224. https://doi.org/10.1007/s10845-010-0425-0.

Resende, M. G., Pitsoulis, L. S., & Pardalos, P. M. (2000). Fortran subroutines for computing approximate solutions of weighted max-sat problems using grasp. *Discrete Applied Mathematics*, *100*, 95–113. doi: https://doi.org/10.1016/S0166-218X(99)00171-7.

Resende, M. G., & Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of Metaheuristics* (pp. 283–319). Boston, MA: Springer US. URL: https://doi.org/10.1007/978-1-4419-1665-5_10.

Resende, M. G., & Ribeiro, C. C. (2016). *Optimization by GRASP: Greedy Randomized Adaptive Search Procedures*. Springer New York. URL: http://doi.org/10.1007/978-1-4939-6530-4.

Rodríguez, J. A., Anjos, M. F., Côté, P., & Desaulniers, G. (2018). MILP formulations for generator maintenance scheduling in hydropower systems. *IEEE Transactions on Power Systems*, *33*, 6171–6180. https://doi.org/10.1109/TPWRS.2018.2833061.

Rodríguez, J. A., Anjos, M. F., Côté, P., & Desaulniers, G. (2021). Accelerating benders decomposition for short-term hydropower maintenance scheduling. *European Journal of Operational Research*, *289*, 240–253. https://doi.org/10.1016/j.ejor.2020.06.041.

Rojanasoonthon, S., & Bard, J. (2005). A GRASP for parallel machine scheduling with time windows. *INFORMS Journal on Computing*, *17*, 32–51. https://doi.org/10.1287/ijoc.1030.0048.

Rojanasoonthon, S., & Bard, J. (2006). A branch-and-price algorithm for parallel machine scheduling with time windows and job priorities. *Naval Research Logistics (NRL)*, *53*, 24–44. https://doi.org/10.1002/nav.20118.

Santos, A. A., Martins, A. X., & Souza, M. J. F. (2022). Modelos heurísticos e meta-heurísticos para resolução de um problema de sequenciamento de ordens de manutenção preventiva de longo prazo. In *Anais do Simpósio Brasileiro de Pesquisa Operacional*. Juiz de Fora, MG, 08 a 11 de novembro: Galoá. Disponível em: https://proceedings.science/sbpo/sbpo-2022/trabalhos/modelos-heuristicos-e-meta-heuristicos-para-a-resolucao-de-um-problema-de-sequen?lang=pt-br.

Schrotenboer, A. H., Ursavas, E., & Vis, I. F. (2020). Mixed integer programming models for planning maintenance at offshore wind farms under uncertainty. *Transportation Research Part C: Emerging Technologies*, *112*, 180–202. https://doi.org/10.1016/j.trc.2019.12.014.

Sedghi, M., Kauppila, O., Bergquist, B., Vanhatalo, E., & Kulahci, M. (2021). A taxonomy of railway track maintenance planning and scheduling: A review and research trends. *Reliability Engineering & System Safety*, *215*, 107827. https://doi.org/10.1016/j.ress.2021.107827.

Shaukat, S., Katscher, M., Wu, C.-L., Delgado, F., & Larrain, H. (2020). Aircraft line maintenance scheduling and optimisation. *Journal of Air Transport Management*, *89*, 101914. https://doi.org/10.1016/j.jairtraman.2020.101914.

Silva, J. M. P., Teixeira, E., & Subramanian, A. (2021). Exact and metaheuristic approaches for identical parallel machine scheduling with a common server and sequence-dependent setup times. *Journal of the Operational Research Society*, *72*, 444–457. doi: 10.1080/01605682.2019.1671153.

Silva, T. A., de Souza, M. C., Saldanha, R. R., & Burke, E. K. (2015). Surgical scheduling with simultaneous employment of specialised human resources. *European Journal of Operational Research*, *245*, 719–730. doi: https://doi.org/10.1016/j.ejor.2015.04.008.

Silva, Y. L. T., Subramanian, A., & Pessoa, A. A. (2018). Exact and heuristic algorithms for order acceptance and scheduling with sequence-dependent setup times. *Computers& Operations Research*, *90*, 142–160. `https://doi.org/10.1016/j.cor.2017.09.006`.

Slotnick, S. A. (2011). Order acceptance and scheduling: A taxonomy and review. *European Journal of Operational Research*, *212*, 1–11. `https://doi.org/10.1016/j.ejor.2010.09.042`.

Slotnick, S. A., & Morton, T. E. (2007). Order acceptance with weighted tardiness. *Computers & Operations Research*, *34*, 3029–3042. `https://doi.org/10.1016/j.cor.2005.11.012`.

Song, T., Xu, K., Li, J., Li, Y., & Tong, Y. (2020). Multi-skill aware task assignment in real-time spatial crowdsourcing. *GeoInformatica*, *24*, 153–173. URL: `https://doi.org/10.1007/s10707-019-00351-4`. doi: 10.1007/s10707-019-00351-4.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. *Mathematical Programming*, *54*, 353–367. `https://doi.org/10.1007/BF01586059`.

Souza Almeida, L., Goerlandt, F., Pelot, R., & Sörensen, K. (2022). A greedy randomized adaptive search procedure (grasp) for the multi-vehicle prize collecting arc routing for connectivity problem. *Computers & Operations Research*, *143*, 105804. URL: `https://www.sciencedirect.com/science/article/pii/S0305054822000910`. doi: https://doi.org/10.1016/j.cor.2022.105804.

Tarhan, I., & Oğuz, C. (2021). Generalized order acceptance and scheduling problem with batch delivery: Models and metaheuristics. *Computers& Operations Research*, *134*, 105414. `https://doi.org/10.1016/j.cor.2021.105414`.

Tarhan, I., & Oğuz, C. (2022). A matheuristic for the generalized order acceptance and scheduling problem. *European Journal of Operational Research*, *299*, 87–103. https://doi.org/10.1016/j.ejor.2021.08.024.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. *Computers& Industrial Engineering*, *58*, 785–800. https://doi.org/10.1016/j.cie.2010.02.012.

Vu, L., Vu, D. M., Hà, M. H., & Nguyen, V.-P. (2022). The two-echelon routing problem with truck and drones. *International Transactions in Operational Research*, *29*, 2968–2994. https://doi.org/10.1111/itor.13052.

Walter, M., & Zimmermann, J. (2012). On a multi-project staffing problem with heterogeneously skilled workers. In D. Klatte, H.-J. Lüthi, & K. Schmedders (Eds.), *Operations Research Proceedings 2011* (pp. 489–494). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wang, H., Alidaee, B., Ortiz, J., & and, W. W. (2021). The multi-skilled multi-period workforce assignment problem. *International Journal of Production Research*, *59*, 5477–5494. URL: https://doi.org/10.1080/00207543.2020.1783009.

Wang, S., & Ye, B. (2019). Exact methods for order acceptance and scheduling on unrelated parallel machines. *Computers& Operations Research*, *104*, 159–173. https://doi.org/10.1016/j.cor.2018.12.016.

Wang, X., Huang, G., Hu, X., & Cheng, T. C. E. (2015). Order acceptance and scheduling on two identical parallel machines. *Journal of the Operational Research Society*, *66*, 1755–1767. https://doi.org/10.1057/jors.2015.3.

Woller, D., Rada, J., & Kulich, M. (2023). The alns metaheuristic for the transmission maintenance scheduling. *Journal of Heuristics*, *29*, 349–382. https://doi.org/10.1007/s10732-023-09514-x.

Wu, G.-H., Cheng, C.-Y., Yang, H.-I., & Chena, C.-T. (2018). An improved water flow-like algorithm for order acceptance and scheduling with identical parallel machines. *Applied Soft Computing*, *71*, 1072–1084. https://doi.org/10.1016/j.asoc.2017.10.015.

Yepes-Borrero, J. C., Perea, F., Villa, F., & Vallada, E. (2023). Flowshop with additional resources during setups: Mathematical models and a grasp algorithm. *Computers & Operations Research*, *154*, 106192. URL: https://www.sciencedirect.com/science/article/pii/S0305054823000564. doi: https://doi.org/10.1016/j.cor.2023.106192.

You, P.-S., & Hsieh, Y.-C. (2021). A heuristic algorithm for medical staff's scheduling problems with multiskills and vacation control. *Sci. Prog.*, *104*, 368504211050301.

Zhang, C., Gao, Y., Yang, L., Kumar, U., & Gao, Z. (2019). Integrated optimization of train scheduling and maintenance planning on high-speed railway corridors. *Omega*, *87*, 86–104. https://doi.org/10.1016/j.omega.2018.08.0.

Zhang, J., Chowdhury, S., Zhang, J., Tong, W., & Messac, A. (2023). Optimal selection of time windows for preventive maintenance of offshore wind farms subject to wake losses. *Wind Energy*, *26*, 1103–1122. https://doi.org/10.1002/we.2815.

Zheng, H.-y., Wang, L., & Zheng, X.-l. (2017). Teaching–learning-based optimization algorithm for multi-skill resource constrained project scheduling problem. *Soft Computing*, *21*, 1537–1548. https://doi.org/10.1007/s00500-015-1866-3.

Çakirgil, S., Yücel, E., & Kuyzu, G. (2020). An integrated solution approach for multi-objective, multi-skill workforce scheduling and routing problems. *Computers & Operations Research*, *118*, 104908. https://doi.org/10.1016/j.cor.2020.104908.

Apêndices

Apêndice A

Complexidade do subproblema de posicionamento de manutenção

Proposição 1. *O subproblema de posicionamento de manutenção (SPM) é um problema $\mathcal{NP}$-difícil.*

Demonstração. Considere um caso específico do SPM, em que a janela de tempo para cada manutenção é igual todo horizonte de planejamento, cada manutenção será executada em diferentes máquinas (o que significa que não haverá sobreposição na programação das manutenção nas máquinas) e todas as equipes tem habilidade para executar todas as manutenções. Além disso, suponha que as penalidades para manutenções não programadas sejam significativamente maiores que um, o que significa que a minimização do número de equipes é irrelevante quando comparada à minimização da penalização para manutenções não programadas.

Portanto, este caso específico do problema consiste em selecionar quais manutenções serão executadas por cada equipe, a fim de minimizar a penalidade total para manutenções não programadas, respeitando a capacidade de cada equipe, dada pelo seu tempo disponível. Este problema pode ser expresso como:

$$\min \sum_{i} w_i(1 - \sum_{k} x_{ik}) \tag{A.1}$$

$$s.t. \sum_{i \in I} p_i x_{ik} \leq h_k \qquad \forall k \in K \tag{A.2}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in I, k \in K \tag{A.3}$$

Esse problema é equivalente ao problema das múltiplas mochilas (PMM), onde cada manutenção corresponde a um item e cada equipe uma mochila. É importante notar que o problema da mochila envolve maximizar o benefício obtido por selecionar

o objeto:

$$\max \sum_i \sum_k w_i x_{ik}.$$

Entretanto, o valor ótimo das variáveis de decisão também pode ser obtido por meio da função objetivo

$$\max \sum_i \sum_k w_i x_{ik} - \sum_i w_i = \sum_i -w_i(1 - \sum_k x_{ik}),$$

como $\sum_i w_i$ é um número constante. Além disso, esta função objetivo é equivalente a

$$\min \sum_i w_i(1 - \sum_k x_{ik}),$$

como o PMM é um $\mathcal{NP}$-difícil (Pisinger, 1999), então o PMM é, também, $\mathcal{NP}$-difícil. □