



**CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS**

Diretoria de Pesquisa e Pós-Graduação

**Programa de Pós-Graduação em
Modelagem Matemática e Computacional**

METAHEURÍSTICAS PARA O PROBLEMA DE SEQUENCIAMENTO EM PROJETOS COM RESTRIÇÃO EM RECURSOS E MÚLTIPLOS MODOS DE EXECUÇÃO

Gustavo Alves Fernandes

Orientador : Prof. Dr. Sérgio Ricardo de Souza

Co-Orientador : Prof. Dr. Moacir Felizardo de França Filho

Belo Horizonte
Julho de 2017



**CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS**

Diretoria de Pesquisa e Pós-Graduação

**Programa de Pós-Graduação em
Modelagem Matemática e Computacional**

METAHEURÍSTICAS PARA O PROBLEMA DE SEQUENCIAMENTO EM PROJETOS COM RESTRIÇÃO EM RECURSOS E MÚLTIPLOS MODOS DE EXECUÇÃO

Dissertação de Mestrado, submetida ao Programa de Pós-Graduação em Modelagem Matemática e Computacional, como parte dos requisitos exigidos para a obtenção do título de Mestre em Modelagem Matemática e Computacional.

Gustavo Alves Fernandes

Orientador : Prof. Dr. Sérgio Ricardo de Souza

Co-Orientador : Prof. Dr. Moacir Felizardo de França Filho

Belo Horizonte
Julho de 2017

F363m Fernandes, Gustavo Alves
Metaheurísticas para o problema de sequenciamento em projetos
com restrição em recursos e múltiplos modos de execução. / Gustavo
Alves Fernandes. – – Belo Horizonte, 2017.
xi, 101 f. : il.

Dissertação (mestrado) – Centro Federal de Educação
Tecnológica de Minas Gerais, Programa de Pós-Graduação em
Modelagem Matemática e Computacional, 2016.

Orientador: Prof. Dr. Sérgio Ricardo de Souza

Coorientador: Prof. Dr. Moacir Felizardo de França Filho

Bibliografia

1. Otimização Combinatória. 2. Programação (Matemática).
3. Heurística. I. Souza, Sérgio Ricardo de. II. Centro Federal de
Educação Tecnológica de Minas Gerais. III. Título

CDD 519.6

AGRADECIMENTOS

O autor gostaria de agradecer:

- aos pais;
- aos professores do MMC, em especial os professores Sérgio Ricardo de Souza e Moacir Felizardo de França Filho;
- aos colegas do MMC os esclarecimentos de pequenas grandes dúvidas;
- à comunidade *Stack Exchange* os esclarecimentos de algumas “picuinhas”;
- às agências CAPES, FAPEMIG e CNPq, o apoio financeiro ao longo do mestrado.

Resumo

O problema conhecido como Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (*Multi-mode Resource Constrained Project Scheduling Problem* - MRCPSP) consiste em sequenciar todas as atividades de um projeto, respeitando a ordem de precedência dessas atividades, que, uma vez iniciadas, não podem ser interrompidas. As atividades possuem várias combinações de modos de execução, que consistem em diferentes combinações de tempo de processamento e consumo de recursos. O objetivo é minimizar o tempo de duração do projeto, encontrando tempos de início para cada atividade do projeto. Por ser um problema NP-Difícil, nesta dissertação são implementadas para sua solução, duas metaheurísticas, *Iterated Local Search* (ILS) e *Clonal Selection Algorithm* (CLONALG), hibridizadas com outras técnicas heurísticas específicas para o MRCPSP. Para cada algoritmo proposto, foram realizados testes estatísticos para verificar influência e tendência dos parâmetros, assim como testes computacionais para comparação entre os algoritmos. Os resultados obtidos para um grupo de instâncias do MRCPSP, entre classes da biblioteca pública PSPLIB apresentaram sucesso em relação a trabalhos presentes na literatura.

PALAVRAS-CHAVE: Sequenciamento em Projetos. *Iterated Local Search*. *Clonal Selection Algorithm*. Metaheurísticas. Problemas de Sequenciamento. Otimização Combinatória.

Abstract

The Multimode Resource Constrained Project Scheduling Problem (MRCPSP) aims to scheduling all activities in a project in which multiple execution modes are available for each activity where there is no preemption among these activities. The optimization problem minimizes the makespan of the project, subject to precedence relations between the activities and resource constraints. To solve this NP-Hard problem we propose the metaheuristics Iterated Local Search (ILS) and Clonal Selection Algorithm (CLONALG) with other specific heuristics techniques for MRCPSP. For each of these metaheuristics we perform statistical tests to verify influence and parameters trend, as well as computational tests to compare the metaheuristics under discussion. Additionally, the performance of the proposed algorithms is tested using benchmark test problems of the project scheduling problem library PSPLIB. Simulation results have shown good performance when compared to results from literature.

KEYWORDS: Project scheduling. Iterated Local Search. Clonal Selection Algorithm. Metaheuristics. Scheduling Problems. Combinatorial optimization.

Lista de Figuras

4.1	Exemplo de projeto em estrutura de rede AoN. Adaptação da Instância j102_2.mm da PSPLIB (cf. Kolisch e Sprecher (1997))	17
4.2	Exemplo de solução para o problema de projeto representado pela Figura 4.1.	21
4.3	Exemplo de sequenciamento obtido pelo SSS a partir da representação dada pela Figura 4.2	23
4.4	Exemplo de aplicação da busca local <i>single-pass</i> MLSI: (a) sequenciamento inicial. (b) sequenciamento final	35
4.5	Exemplo de sequenciamento obtido pela heurística <i>multi-pass</i> MFBI: a) sequenciamento <i>forward</i> inicial; b) sequenciamento <i>backward</i> ; c) sequenciamento <i>forward</i> final.	38
5.1	Exemplo de solução inicial	45
5.2	Sequenciamento obtido pelo Algoritmo 7	45
5.3	Movimento N_1 aplicado à solução corrente s levando a um vizinho s' .	47
5.4	Movimento N_2 aplicado à solução corrente s levando a um vizinho s' .	47
5.5	Movimento N_3 aplicado à s levando a um vizinho s'	48
5.6	Número de modificações em função de k e MAX_MOD	50
6.1	Número de modificações Θ em função dos níveis de perturbação $k = 1, \dots, K$ para máximas perturbações 0.3 e 0.9, de acordo com a Equação 5.4, sobre a classe J20.	55
6.2	Tendência dos parâmetros do ILS-MRCPSP ao variar o nível.	56
6.3	Boxplot para os níveis do parâmetro K em relação ao GAP	58
6.4	Boxplot para os níveis do parâmetro MAX_MOD em relação ao GAP .	58
6.5	Interação entre os parâmetros K e MAX_MOD pela variação de cada nível	59
6.6	Diferenças entre as médias dos níveis do parâmetro K	60
6.7	Diferenças entre as médias dos níveis do parâmetro MAX_MOD . .	60
6.8	Tendência dos parâmetros do CLONALG-MRCPSP ao variar o nível. .	62
6.9	Boxplot para os níveis de cada parâmetro.	63
6.10	Interação entre os parâmetros POP , $N1$, $N2$ e β por meio da variação de cada nível.	64
6.11	Diferença entre as médias dos níveis de cada parâmetro POP , $N1$, $N2$ e β	66
6.12	Diferença entre os algoritmos para determinação das melhores soluções conhecidas e GAP por instância solucionada.	68

6.13	Boxplot para a diferença entre os resultados de GAP obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.	69
6.14	Boxplot para a diferença entre os ótimos encontrados obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.	69
6.15	Boxplot para a diferença entre os tempos (segundos) obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.	70
B.1	Boxplot para comparações entre as combinações ($POP : N1$)	88
B.2	Boxplot para comparações entre as combinações ($POP : N2$)	89
B.3	Boxplot para comparações entre as combinações ($N1 : N2$)	90
B.4	Boxplot para comparações entre as combinações ($POP : \beta$)	91
B.5	Boxplot para comparações entre as combinações ($N1 : \beta$)	92
B.6	Boxplot para comparações entre as combinações ($N2 : \beta$)	93
B.7	Boxplot para comparações entre as combinações ($POP : N1 : \beta$)	94
B.8	Boxplot para comparações entre as combinações ($POP : N2 : \beta$)	95
C.1	Diferença para as médias das comparações entre os níveis dos parâmetros POP e $N1$. As comparações destacadas em vermelho apresentam diferenças estatísticas com 95% de confiança	96
C.2	Diferença para as médias das comparações entre os níveis dos parâmetros POP e $N2$	97
C.3	Diferença para as médias das comparações entre os níveis dos parâmetros $N1$ e $N2$	98
C.4	Diferença para as médias das comparações entre os níveis dos parâmetros POP e β	99
C.5	Diferença para as médias das comparações entre os níveis dos parâmetros $N1$ e β	100
C.6	Diferença para as médias das comparações entre os níveis dos parâmetros $N2$ e β	101

Lista de Tabelas

2.1	Notação para o problema MRCPSP	5
3.1	Heurísticas adotadas na resolução do MRCPSP com minimização do makespan	15
4.1	Modos de execução para as atividades no exemplo de projeto apresentado pela Figura 4.1. Adaptação da Instância j102_2.mm da PSPLIB.	18
4.2	Exemplo de execução do Algoritmo 1 sobre o problema de projeto apresentado pela Figura 4.1	20
4.3	Exemplo de execução passo-a-passo do Algoritmo SSS.	23
4.4	Exemplo de execução passo-a-passo do Algoritmo PSS	25
4.5	Exemplos de regras de prioridade para modos de execução	27
4.6	Exemplos de Regras de prioridade baseadas em atividades	29
4.7	Exemplos de Regras de prioridade baseadas na representação de rede	30
4.8	Exemplos de Regras de prioridade baseadas no caminho crítico	30
4.9	Exemplos de Regras de prioridade baseadas em recursos	30
4.10	Exemplos de Regras de prioridade compostas	30
5.1	Regras de prioridade MNM para modos de execução	42
5.2	Cálculo dos valores de prioridade e probabilidade tendenciosa para a regra SFM e MNM, por meio do método BRS. As células marcadas correspondem aos modos selecionados para cada atividade pela regra SFM.	44
5.3	Cálculo dos valores de probabilidade para a regra LFT pelo método RBBRS. As atividades j^* são aquelas escolhidas dentre todas as atividades disponíveis em D_n	45
6.1	Classes de instâncias PSPLIB	52
6.2	Ordem dos experimentos para cada combinação de níveis dos parâmetros	54
6.3	Valores GAP_{avr}^1 e GAP_{avr}^2 obtidos para cada combinação de parâmetro	56
6.4	Respostas entre as médias dos níveis de parâmetros	57
6.5	Tabela ANOVA para $\alpha = 0.05$	59
6.6	Ordem dos experimentos para cada combinação de valores dos parâmetros	61
6.7	Respostas entre as médias das combinações de níveis de parâmetros	62
6.8	Tabela ANOVA para $\alpha = 0.05$	65
6.9	Comparação entre CLONALG-MRCPSP e ILS-MRCPSP para 5000 soluções geradas	68

6.10	Valores de $GAP_{avr}(\%)$ obtidos para as classes da PSPLIB.	71
6.11	Ótimos $(\%)$ obtidos para as classes da PSPLIB.	71
A.1	Média do valores de GAP para cada combinação de parâmetro do algoritmo CLONALG	85

Sumário

1	Introdução	1
1.1	Objetivos: Geral e Específicos	2
1.2	Sobre a Metodologia	3
1.3	Organização da Dissertação	3
2	Caracterização do Problema	4
2.1	Sequenciamento em Projetos com Restrição em Recursos	5
2.2	Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução	7
3	Trabalhos Relacionados	10
3.1	Abordagens via <i>Branch-and-Bound</i>	10
3.2	Abordagens via (Meta)heurísticas	12
4	Modelos e Algoritmos	17
4.1	Representação de Projetos	17
4.2	Método do Caminho Crítico	18
4.3	Representação das Soluções	20
4.4	Esquemas de Geração de Sequenciamento	21
4.4.1	Método Serial	21
4.4.2	O método Paralelo	24
4.5	Heurísticas Construtivas	26
4.5.1	Regras de Prioridade	26
4.5.2	Métodos multi-pass	31
4.6	Heurísticas de Refinamento	33
4.6.1	<i>Multi-mode Left Shift Improvement</i> - MLSI	33
4.6.2	<i>Multi-mode forward/backward improvement</i> - MFBI	35
4.7	<i>Iterated Local Search</i> - ILS	39
4.8	<i>Clonal Selection Algorithm</i>	39
5	Algoritmos para a resolução do MRCPSP	42
5.1	Construção de uma solução e esquemas de sequenciamento	42
5.2	Estruturas de Vizinhança	46
5.3	Função de Avaliação	48
5.4	Adaptação da metaheurística ILS para o MRCPSP	49
5.5	Adaptação da metaheurística CLONALG para o MRCPSP	51

6	Resultados Computacionais	52
6.1	Descrição dos Experimentos	52
6.2	Análise de parâmetros do ILS-MRCPSP	54
6.3	Análise de parâmetros do CLONALG-MRCPSP	61
6.4	Comparação entre ILS-MRCPSP e CLONALG-MRCPSP	67
6.5	Comparação com a literatura	70
7	Conclusões	73
	Referências	75
	Apêndice A Tabela com respostas obtidas pelas combinações de parâmetros do algoritmo CLONALG-MRCPSP	85
	Apêndice B Boxplots para combinações de ordem 2 e 3 entre os parâmetros do algoritmo CLONALG-MRCPSP	88
	Apêndice C Teste de Tukey para combinações de ordem 2 entre os parâmetros do algoritmo CLONALG-MRCPSP	96

Capítulo 1

Introdução

Nos dias atuais, o campo de gerenciamento de projetos possui grande importância e relevância nas empresas. Para [Demeulemeester \(2002\)](#), em ambientes competitivos é desejável entregar produtos com boa qualidade e baixo custo. A ISO (*International Organization for Standardization*) 8402 define *projeto* como um único processo, consistindo de um conjunto controlado de atividades com datas de início e fim, procurando atingir um objetivo, conforme requerimentos específicos, incluindo restrições de tempo, custo e recursos.

Na atualidade, projetos variados se apresentam em diferentes formas, como aquelas destacadas em [Demeulemeester \(2002\)](#): construção civil; abertura de um novo empreendimento; modelagem de novas aeronaves ou automóveis; desenvolvimento e implementação de software; realocação de imóveis; introdução de um novo produto no mercado; produção e direção de um filme; planejamento do casamento; escrita de um livro, artigo ou trabalho universitário; dentre outros.

[Demeulemeester \(2002\)](#) afirma que gerenciamento de projeto, basicamente, considera planejamento, sequenciamento e controle das atividades do projeto para alcançar satisfação em custos e objetivos de tempo para um dado escopo de trabalho, enquanto usando eficiente e eficazmente recursos. O sequenciamento em projetos especifica a precedência entre as atividades e a disponibilidade de recursos necessários a cada período, ao longo da execução do projeto, para ordenar as atividades e executá-las.

Um projeto consiste em um número finito de atividades que devem ser executadas de acordo com um conjunto de restrições de precedências. Cada atividade possui uma duração e, normalmente consome recursos. Recursos podem ser de tipos diferentes, como recursos financeiros, mão de obra, maquinário, equipamentos, materiais, energia e espaço. No gerenciamento de projetos, é necessário definir quais recursos serão utilizados na execução do projeto, a disponibilidade de cada um deles e, também, a necessidade de estimar o consumo dos recursos pelas atividades.

Uma distinção comum entre categorias de recursos, apresentada em [Błażewicz \(1986\)](#), é dada pelas categorias de recursos renováveis, não renováveis e recursos duplamente restritos. Recursos renováveis estão disponíveis de período a período, ou seja, o total é renovado a cada período, à medida em que as tarefas são concluídas. Somente o total de recurso usado a cada instante é restrito. Típicos exemplos de recursos renováveis incluem mão de obra, maquinário, ferramentas, equipamentos e espaço. Já para recursos não renováveis há uma quantidade disponível inicial, que

vai sendo consumida a cada instante, sem a possibilidade de reposição. Um exemplo simples de recursos desse tipo é o capital. Outros exemplos são matéria prima e combustível. Recursos duplamente limitados são restritos por período, assim como por todo o projeto. Um exemplo é homem-hora por dia, em que, a quantidade total de trabalhadores é disponível em todos os dias, contudo, existe a restrição do número total de horas trabalhadas por dia para cada trabalhador.

É destacado em [Demeulemeester \(2002\)](#) que vários problemas relacionados ao contexto de sequenciamento e operações de produção compartilham características que os tornam passíveis a técnicas desenvolvidas no contexto de sequenciamento em projetos. [Demeulemeester \(2002\)](#) acrescenta que vários problemas de sequenciamento estudados, como *Single Machine Problems*, *Parallel Machine Problems*, *Flow Shops* e *Job Shops* são modelados como variações do problema clássico conhecido como Sequenciamento em Projetos com Restrição em Recursos (*Resource Constrained Project Scheduling Problem - RCPSP*), que consiste em sequenciar as atividades, respeitando a precedência entre elas, e também, o consumo de recursos renováveis. [Hartmann e Briskorn \(2010\)](#) discutem variações do RCPSP, sejam em objetivos, recursos e atividades, com a justificativa de que o RCPSP não abrange todas as situações que podem ocorrer na prática.

Esta dissertação aborda uma generalização do RCPSP destacada pela literatura e conhecida como Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (*Multi-mode Resource Constrained Project Scheduling Problem - MRCPSP*), e propõe a aplicação de duas metaheurísticas para a resolução do MRCPSP.

1.1 Objetivos: Geral e Específicos

O objetivo geral desta dissertação é analisar o comportamento dos algoritmos *Iterated Local Search* (ILS) e *Clonal Selection Algorithm* (CLONALG) na resolução do MRCPSP. Para cumprir essa meta, é necessário alcançar os seguintes objetivos específicos:

- (i) Implementação dos métodos propostos;
- (ii) Aplicar os métodos em instâncias conhecidas pela literatura;
- (iii) Realizar análises estatísticas e empíricas dos resultados produzidos pelos algoritmos;
- (iv) Realizar comparações dos resultados obtidos em relação aos trabalhos presentes na literatura.

Em relação ao item (i), esta dissertação apresenta a implementação das metaheurísticas *Iterated Local Search* (ILS) e CLONALG, em conjunto com heurísticas específicas, para a solução do MRCPSP. Para o item (ii), são utilizadas as instâncias existentes na PSPLIB, descritos em [Kolisch e Sprecher \(1997\)](#), para resolução do MRCPSP com minimização do *makespan*. Essa biblioteca contém um conjunto de classes para teste com instâncias agrupadas por número de atividades, além de resultados ótimos e heurísticos encontrados. Sobre o item (iii), as análises estáticas

são realizadas por testes ANOVA e Tukey, por meio do *software* R para verificar tendências dos parâmetros dos algoritmos ILS e CLONALG. Finalmente, quanto ao item (iv), são discutidos os resultados e relevâncias entre os algoritmos presentes nesta dissertação e os algoritmos da literatura, para resolução do MRCPSP.

1.2 Sobre a Metodologia

Para solucionar as classes de problemas da PSPLIB relacionadas ao MRCPSP com minimização do *makespan*, serão agregados procedimentos específicos do problema em discussão às metaheurísticas ILS e CLONALG. Esses procedimentos possibilitam explorar características do MRCPSP com o objetivo de agregá-las aos algoritmos ILS e CLONALG. Então, são implementadas uma heurística construtiva baseada em regras de prioridade, assim como uma heurística de busca local destacada pela literatura. Também são exploradas estruturas de vizinhança com o objetivo de realizar modificações na representação computacional de uma solução para o problema em discussão.

1.3 Organização da Dissertação

A dissertação está organizada da seguinte maneira. O Capítulo 2 apresenta a descrição do problema de interesse, bem como sua formulação recorrente na literatura e a notação utilizada nesta dissertação. No Capítulo 3 é apresentada a revisão literária sobre o MRCPSP. São discutidos os métodos exatos com ênfase na minimização do *makespan* para o MRCPSP e, também, são apresentadas discussões sobre as principais (meta)heurísticas na resolução do MRCPSP. No Capítulo 4 é apresentado o referencial teórico abordando a descrição das heurísticas e metaheurísticas utilizadas pela literatura para a solução do problema objeto de interesse. O Capítulo 5 apresenta as implementações heurísticas propostas nessa dissertação para resolução do MRCPSP. Já no Capítulo 6, são apresentados os resultados obtidos por implementações realizadas, assim como comparações com as melhores metaheurísticas presentes na literatura, sendo que instâncias públicas foram utilizadas para os testes computacionais. Finalmente, o Capítulo 7 apresenta conclusões sobre o tema em discussão nesta dissertação.

Capítulo 2

Caracterização do Problema

De acordo com [Lawler \(1976\)](#), análise combinatória é o estudo do arranjo, agrupamento, ordenação, ou seleção de objetos, usualmente números discretos e finitos. Pesquisadores em análise combinatória, como exposto por [Lawler \(1976\)](#), estão preocupados com questões de existência ou enumeração. Ou seja, existe um tipo particular de arranjo? Ou, quantos desses arranjos existem? Mas há uma linha na área da investigação combinatória com grande destaque, em que a questão a ser respondida não é “*Existe um certo tipo de arranjo?*” ou “*Quantos desses arranjos existem?*”, mas sim “*Qual é o melhor arranjo?*”. Assim, o que importa nessa investigação é encontrar o arranjo ótimo diante das várias possibilidades de um conjunto finito. Essa linha de investigação é conhecida como Otimização Combinatória.

O objetivo do processo de otimização é encontrar uma solução viável, que represente um mínimo (ou máximo) global para uma determinada função objetivo, dentro do domínio formado pelas restrições do problema a ser otimizado. Porém, encontrar uma solução que satisfaça todas as restrições e que, simultaneamente, atenda a todos os objetivos pode não ser uma tarefa fácil, sendo, em vários casos, até mesmo improvável.

É crescente o número de problemas de Otimização Combinatória conhecidos que vêm sendo pesquisados ao longo dos anos, devido ao potencial de aplicação destes problemas no mundo real. Em [Jünger et al. \(2009\)](#), é apresentada uma coleção de artigos com contribuição expressiva em relação a problemas de otimização entre os anos de 1958 e 2008.

[Papadimitriou e Steiglitz \(1998\)](#) descrevem problemas de otimização como um conjunto I de instâncias, tal que uma instância de um problema de otimização é um par (F, c) , sendo F o conjunto de pontos viáveis e c uma função de custo, dado que $c : F \rightarrow \mathbb{R}$. O objetivo é, então, encontrar $f \in F$ tal que $c(f) \leq c(y), \forall y \in F$, caso deseja-se a minimização de c . Esse ponto f é chamado de ótimo global para a instância dada ou simplesmente solução ótima. Assim, uma instância é representada por dados de entrada e possui informações suficientes para obter uma solução presente nesses dados.

Para a apresentação do problema de otimização discutido na presente dissertação, será adotada a notação apresentada pela Tabela [2.1](#).

Tabela 2.1: Notação para o problema MRCPSP

Símbolo	Definição
J	Conjunto de atividades (ou tarefas) presentes em um projeto
$j \in J$	Atividade j presente em um projeto
M_j	Conjunto de modos de execução da atividade j
$m \in M_j$	Modo m de execução da atividade j
$\mathcal{P}_j$	Conjunto de predecessores da atividade j
$\mathcal{S}_j$	Conjunto de sucessores da atividade j
d_{jm}	Tempo de duração de uma atividade j no modo m
K	Quantidade de recursos presentes no projeto
$k = 1, \dots, K$	Tipo de um recurso k presente no projeto
R_k^ρ	Quantidade disponível do recurso renovável do tipo k
R_k^ν	Quantidade disponível do recurso não renovável do tipo k
r_{jmk}^ρ	Quantidade do recurso renovável k consumida pela atividade j no modo m
r_{jmk}^ν	Quantidade do recurso não renovável k consumida pela atividade j no modo m
ST_j	Tempo de início de uma atividade j
FT_j	Tempo de término de uma atividade j
EST_j	Tempo de início mais cedo de uma atividade j
EFT_j	Tempo de término mais cedo de uma atividade j
LST_j	Tempo de início mais tardio de uma atividade j
LFT_j	Tempo de término mais tardio de uma atividade j
T	Horizonte do projeto
C_{max}	duração do projeto ou <i>makespan</i>

2.1 Sequenciamento em Projetos com Restrição em Recursos

O problema conhecido como Sequenciamento em Projetos com Restrição em Recursos (*Resource Constrained Project Scheduling Problem - RCPSP*) é combinatório e é caracterizado da seguinte forma. Seja um projeto com $|J|$ atividades, de modo que cada atividade $j \in J$ deve ser processada sem interrupção, respeitando-se a relação de precedência entre as mesmas, i.e., cada atividade j pode ser iniciada somente após todas as suas predecessoras em $\mathcal{P}_j$ estarem finalizadas. As atividades $j = 1$ e $j = |J|$ são fictícias, de modo que $j = 1$ deve ser a primeira atividade a ser iniciada sem algum predecessor e $j = |J|$ deve ser a última atividade com nenhum sucessor. Para cada atividade j do projeto, existe um tempo de processamento d_j e uma demanda de recursos renováveis r_{kj}^ρ para essa atividade. Durante a execução do projeto, para cada período t , existe uma quantidade fixa de recursos renováveis R_k^ρ . Deve-se ressaltar que as atividades fictícias possuem tempo de duração e consumo de recursos iguais a zero. Considera-se que todos os parâmetros são valores inteiros não negativos. Então, o objetivo do RCPSP é encontrar uma combinação de sequência entre as atividades, resultando em um sequenciamento com tempos de início, definidos de modo a respeitar as relações de precedência e disponibilidade de recursos, tal que o *makespan* (duração total do projeto) seja minimizado. A variável de decisão x_{jt} é igual a 1 se a atividade j é finalizada no período t . Caso contrário,

x_{jt} é igual a 0.

A formulação matemática recorrente na literatura para o RCPSP foi apresentada por Pritsker et al. (1969) e é dada por:

$$\min \sum_{t=EFT_{|J|}}^{LFT_{|J|}} tx_{|J|t} \quad (2.1)$$

s.a.

$$\sum_{t=EFT_j}^{LFT_j} x_{jt} = 1, \quad j \in J \quad (2.1)$$

$$\sum_{t=EFT_j}^{LFT_j} (t - d_j)x_{jt} - \sum_{t=EFT_i}^{LFT_i} tx_{it} \geq 0, \quad j \in J; i \in \mathcal{P}_j \quad (2.2)$$

$$\sum_{j \in J} \sum_{s=t}^{t+d_j} r_{jk}^{\rho} x_{js} \leq R_{kt}^{\rho}, \quad k = 1, \dots, K; t = 1, \dots, LST_{|J|} \quad (2.3)$$

$$x_{jt} \in \{0, 1\}, \quad j \in J; t = EFT_j, \dots, LFT_j \quad (2.4)$$

Por essa formulação, a expressão (2.1) apresenta o objetivo de minimizar o término da última atividade, representada por $x_{|J|t}$. A expressão (2.1) representa a atribuição de um único tempo de término por atividade. A expressão (2.2) apresenta a relação de precedência entre atividades. A expressão (2.3) assegura que a disponibilidade de recursos renováveis por período não é violada. Finalmente, a expressão (2.4) indica que as variáveis de interesse x_{jt} são binárias. Os valores EST_j , LST_j , EFT_j e LFT_j são obtidos pelo método do caminho crítico, descrito, assim como os valores EST_j , LST_j , EFT_j e LFT_j , no Capítulo 4.

Embora o RCPSP possua um bom e aplicável modelo, esse problema não cobre todas as situações que ocorrem na prática, conforme aponta Hartmann e Briskorn (2010). Consequentemente, diversas variações do RCPSP foram propostas, como:

- Sequenciamento preemptivo;
- Variação da requisição de recursos no tempo por cada atividade;
- Tempo de preparo para cada atividade ser iniciada (*setup time*);
- Múltiplos modos de execução para cada atividade, sendo que cada modo consiste de tempo de duração e consumo de recursos;
- Datas de *release* para cada atividade ser iniciada, assim como *deadlines* para término;
- Acréscimo de recursos não renováveis ou recursos duplamente restritos;
- Recursos com disponibilidade contínua, ao contrário dos recursos discretos já comentados, como energia e materiais líquidos;
- Recursos dedicados a cada atividade por período;

- Capacidade de recursos variando com o tempo;
- Múltiplos projetos, em que cada projeto em separado possui um conjunto de atividades.

Recentemente, um novo problema foi proposto no evento MISTA (*Multidisciplinary International Scheduling Conference - Theory & Applications*), em sua edição de 2013, para o *MISTA 2013 challenge*, que é uma competição proposta com o objetivo de estimular pesquisas em problemas de sequenciamento. Esse novo problema é identificado como Sequenciamento em Múltiplos Projetos com Restrição em Recursos e Múltiplos Modos de Execução, sendo especificado em [Wauters et al. \(2014\)](#).

Revisões que apresentam, detalhadamente, variações relacionadas a atividades, recursos e objetivos são apresentadas em [Kramer e Hwang \(1991\)](#), [Demeulemeester \(2002\)](#), [Hartmann e Briskorn \(2010\)](#), [Węglarz et al. \(2011\)](#), [Lombardi e Milano \(2012\)](#) e [Wu et al. \(2014\)](#).

A variação do RCPSP a ser considerada na presente dissertação será o Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução.

2.2 Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução

O problema conhecido como Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (*Multi-mode Resource Constrained Project Scheduling Problem - MRCPSP*) possui uma caracterização semelhante ao RCPSP por ser uma generalização do último. A diferença está na existência de modos de execução nas quais cada atividade $j \in J$ do projeto deve ser executada em um modo $m \in M_j$ de execução, que estabelece o tempo de processamento e a demanda de recursos para a atividade. As atividades fictícias possuem um único modo de execução, com tempo de duração e consumo de recursos nulos. O número de modos em que uma atividade j pode ser executada é dado por $|M_j|$ e o tempo de processamento da atividade j no modo m é denotado por d_{jm} . Há, também, a adição de recursos não renováveis, ao contrário do RCPSP, que possui somente recursos renováveis. Então, existem as quantidades R_k^ρ e R_k^ν . As quantidades r_{jmk}^ρ e r_{jmk}^ν definem, respectivamente, o consumo de recursos renováveis e não renováveis do tipo k no modo m pela atividade j . Os recursos não renováveis possuem uma quantidade fixa e disponível durante toda a execução do projeto, mas, ao contrário dos recursos renováveis, o primeiro não restabelece sua capacidade para cada período. Considera-se que todos os parâmetros são valores inteiros não negativos. Portanto, o objetivo do MRCPSP é encontrar uma combinação dos modos de execução para todas as atividades, assim como uma combinação de sequências entre essas atividades, resultando em um sequenciamento com tempo de início e consumo de recursos para cada atividade, de maneira que o *makespan* seja minimizado. A variável de decisão é denotada por x_{jmt} , sendo igual a 1 se a atividade j é executada no modo m e possui término no período t . Caso contrário, x_{jmt} é igual a 0.

A formulação matemática recorrente na literatura e apresentada por Talbot (1982) para o MRCPSP é dada por:

$$\min \sum_{m \in M_J} \sum_{t=EFT_{|J|}}^{LFT_{|J|}} tx_{|J|mt} \quad (2.5)$$

s.a.

$$\sum_{m \in M_J} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1, j \in J \quad (2.5)$$

$$\sum_{m \in M_J} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm})x_{jmt} - \sum_{m \in M_J} \sum_{t=EFT_i}^{LFT_i} tx_{imt} \geq 0, j \in J; i \in \mathcal{P}_j \quad (2.6)$$

$$\sum_{j \in J} \sum_{m \in M_J} \sum_{s=t}^{t+d_{jm}} r_{jmk}^\rho x_{jms} \leq R_{kt}^\rho, k = 1, \dots, K^\rho; t = 1, \dots, LST_{|J|} \quad (2.7)$$

$$\sum_{j \in J} \sum_{m \in M_J} \sum_{t=EFT_j}^{LFT_j} r_{jmk}^\nu x_{jmt} \leq R_k^\nu, k = 1, \dots, K^\nu \quad (2.8)$$

$$x_{jmt} \in \{0, 1\} (j \in J; m \in M_j; t = EFT_j, \dots, LFT_j) \quad (2.9)$$

Nessa formulação, a expressão (2.5) apresenta o objetivo de minimizar o término da última atividade; já a expressão (2.5) representa a execução de um modo por atividade, assim como a atribuição de um único tempo de término; a expressão (2.6) apresenta a relação de precedência entre atividades; a expressão (2.7) assegura que a disponibilidade de recursos renováveis por período não seja violada; a expressão (2.8) corresponde à restrição de recursos não renováveis; finalmente, a expressão (2.9) mostra a característica binária da variável de decisão x_{jmt} . Os valores EST_j , LST_j , EFT_j e LFT_j são obtidos de maneira semelhante ao RCPSP, após a atribuição do modo com menor duração, para cada atividade.

Percebe-se que a formulação apresentada para o RCPSP também aparece na formulação para o MRCPSP e que, então, o MRCPSP apresenta dois subproblemas:

- (i) selecionar um conjunto de modos de execução que seja viável, isto é, que respeite as restrições associadas ao consumo de recursos não renováveis; e após,
- (ii) sequenciar todas as atividades de acordo com a atribuição de modos, respeitando as restrições de recursos renováveis e de precedência.

De acordo com Blazewicz et al. (1983), encontrar um sequenciamento ótimo para o RCPSP é um problema NP-Difícil e, por ser uma generalização do RCPSP, o MRCPSP também é um problema NP-Difícil. Já Drexel e Gruenewald (1993) apontam que nem sempre é possível encontrar soluções viáveis para o MRCPSP. Foi demonstrado por Kolisch e Drexel (1997) que obter uma solução viável (uma atribuição viável de modos para cada atividade) para o MRCPSP, com pelo menos dois tipos de recursos não renováveis e pelo menos dois modos de execução, é um problema NP-Completo. Conforme apontam Sprecher e Drexel (1998), projetos com mais de vinte atividades, com, no mínimo, três modos de execução, não podem ser

solucionados na otimalidade por métodos exatos em tempo aceitável e sugerem que métodos exatos truncados ou heurísticas são boas abordagens para problemas de maior número de modos de execução e maior número de atividades.

Devido à complexidade computacional do MRCPSP, assim como sua aplicação em situações reais, vários autores propuseram métodos diversos para a resolução desse problema. Uma revisão bibliográfica é apresentada no capítulo 3 a seguir.

Capítulo 3

Trabalhos Relacionados

Várias abordagens são encontradas na literatura para a resolução do MRCPSP. Os métodos via *branch-and-bound* se destacam como soluções exatas na resolução do MRCPSP. Já as metaheurísticas de trajetória e populacionais recebem destaque como técnicas heurísticas para a resolução do MRCPSP.

3.1 Abordagens via *Branch-and-Bound*

Um algoritmo é dito exato se encontra a solução ótima de um problema, de acordo com condições matemáticas de otimalidade do mesmo. Dentre os algoritmos exatos, são encontrados os algoritmos de enumeração e de pesquisa. Como descrito em [Papadimitriou e Steiglitz \(1998\)](#), o método *Branch-and-Bound* (B&B) é uma enumeração inteligente das soluções viáveis de um problema de Otimização Combinatória. Os autores destacam a qualificação **inteligente** pela razão de que não é simples enumerar completamente todas as possíveis soluções. O termo *branch* se refere ao particionamento do espaço de soluções; já o termo *bound* refere-se a limites superiores ou inferiores usados para construir a prova de otimalidade, sem a necessidade de realizar uma busca exaustiva no espaço de soluções. Devido à dificuldade apresentada naturalmente pelo MRCPSP, poucos pesquisadores apresentaram algoritmos B&B na literatura para a resolução desse problema.

[Patterson e Weglarz \(1989\)](#) propuseram um algoritmo B&B conhecido como Árvore de Precedência (*Precedence Tree*). Árvore de Precedência permite uma enumeração sistemática de atividades, atribuindo modos e tempos de início (ou tempos de término) para as atividades. Em um nível l da árvore, uma atividade elegível j_l é identificada e um modo m_{j_l} atribuído, e, em seguida, é computado um tempo de início viável s_{j_l} , que não é menor que o tempo de início atribuído no nível anterior da árvore. Então, o método se desloca para o próximo nó, até que a atividade fictícia final seja elegível e, finalmente, um sequenciamento é obtido. Um *backtracking* é realizado ao longo da árvore para gerar soluções alternativas e o B&B finaliza sua execução assim que o *backtracking* alcança o nível $l = 0$. Ou seja, diversos sequenciamentos são construídos, sendo que a raiz da árvore é a primeira atividade (fictícia), uma folha da árvore é a última atividade (fictícia) a ser sequenciada e cada nó interno é uma atividade real. Assim, de acordo com [Patterson e Weglarz \(1989\)](#), um caminho da raiz até uma folha representa um sequenciamento, respeitando sempre

a restrição de precedência. Em seguida, Sprecher (1994) e Sprecher e Drexl (1998) aprimoraram a abordagem, incluindo outros critérios de poda (*bound*).

Já Sprecher et al. (1997) apresentaram uma nova abordagem B&B, introduzindo o conceito de Alternativas de Modo e Atraso (*Mode and Delay Alternatives*). Esta abordagem também realiza a construção de um sequenciamento ao longo de uma árvore. Porém, cada nó representa um instante de tempo no qual as atividades elegíveis são sequenciadas (atribuição de tempo de início) temporariamente. Se alguma restrição de recursos for violada, será necessário decidir qual(uais) atividade(s) elegível(eis) sofrerá(ão) atraso(s) em seu(s) tempo(s) de início, ou seja, essas atividades serão removidas do conjunto de atividades em progresso e estarão disponíveis para o próximo ponto de decisão (ou nó da árvore).

Posteriormente, utilizando o conceito de alternativas de modo e atraso, Hartmann e Drexl (1998) criaram o conceito de Alternativas de Modo e Extensão (*Mode and Extension Alternatives*). Igualmente ao conceito anterior, nessa abordagem um nó representa um instante de tempo no qual as atividades elegíveis são colocadas em progresso. Contudo, o sequenciamento parcial é estendido por meio da atribuição de tempos de início para um subconjunto de atividades elegíveis, de tal maneira a não violar qualquer restrição de recursos, ou seja, somente um subconjunto das atividades elegíveis será de fato iniciado naquele instante de tempo. Essas três abordagens B&B podem ser encontradas, detalhadamente, com comparações entre si, em Hartmann e Drexl (1998).

A última abordagem exata para resolução do MRCPSP com minimização do *makespan*, conhecida pelo autor desta dissertação, é devida a Zhu et al. (2006), no qual é apresentado um algoritmo *Branch-and-Cut* para o MRCPSP. Os recursos são parcialmente renováveis, de modo que se considera os recursos renováveis e não renováveis como casos especiais. Neste último trabalho, a partir do modelo linear inteiro apresentado pela formulação para o MRCPSP, é utilizada uma relaxação para obter limites inferiores para cada nó da árvore de enumeração. Se o nó possui um valor fracionado em relação ao sequenciamento parcial, então o algoritmo tenta encontrar desigualdades (ou cortes) válidas, que são violadas pela solução fracionada, porém satisfeitas por todas as soluções viáveis e inteiras representadas por esse nó. Se nenhuma desigualdade é encontrada, é realizado um *branching* para gerar outros nós na árvore. Foi utilizado um algoritmo genético para obter soluções iniciais. Os autores reportaram que determinaram soluções com valores de *makespan* para 5 instâncias, da classe J30 da PSPLIB (cf. Kolisch e Sprecher (1997)), melhores do que os anteriormente conhecidos para essas instâncias. Os autores também reportam tempo na determinação do sequenciamento ótimo, melhores do que os encontrados por Sprecher e Drexl (1998). Entretanto, verificando a diferença da capacidade de processamento em ambos trabalhos, o último apresenta uma abordagem enumerativa mais rápida. Destaca-se também que, no trabalho de Zhu et al. (2006), foram determinados ótimos para 506 das 552 instâncias da classe J30, para o critério de parada dado por 3600 segundos. Essa classe contém as maiores instâncias na PSPLIB para o MRCPSP.

Devido às dificuldades encontradas por todos esses autores na resolução de instâncias em larga dimensão para determinação do sequenciamento ótimo, abordagens via (meta)heurísticas foram apresentadas para a resolução do MRCPSP.

3.2 Abordagens via (Meta)heurísticas

As primeiras heurísticas para resolução do MRCPSP, considerando recursos renováveis e não renováveis, são encontradas em [Talbot \(1982\)](#), que emprega um B&B truncado com utilização de regras de prioridade e um sequenciador em serial (Regras de prioridade e Sequenciadores serão detalhados no Capítulo 4). [Drexel e Gruenewald \(1993\)](#) apresentam a abordagem *Regret-Based Biased Random Sampling*, detalhada no Capítulo 4, com um sequenciador em serial. Já [Özdamar e Ulusoy \(1994\)](#) propuseram um algoritmo aproximado utilizando um sequenciador em paralelo. [Kolisch e Drexel \(1996\)](#) apresentaram uma abordagem *Multi-Pass Regret-Based Biased Random Sampling* adaptativa, sendo que o método, em tempo de busca, altera entre dois sequenciadores, serial e paralelo, de acordo com a instância. [Kolisch e Drexel \(1997\)](#) são os primeiros autores a apresentarem uma heurística de busca local, que encontra soluções ótimas ou aproximadas do ótimo para instâncias de maiores dimensões. Outras heurísticas clássicas para o MRCPSP, que não levam em consideração os recursos não renováveis, foram apresentadas em [Elmaghraby \(1977\)](#), [Boctor \(1993\)](#) e [Boctor \(1996\)](#).

Considerando os últimos 15 anos, diversas metaheurísticas foram adaptadas para resolução do MRCPSP. Em [Jozefowska et al. \(2001\)](#), é proposta uma implementação da metaheurística *Simulated Annealing* (SA), fazendo a busca alternadamente em vizinhanças baseadas em atividades e modos de execução. Em [Bouleimen e Lecocq \(2003\)](#), também é apresentado uma implementação de SA, tendo a busca local em dois estágios. No primeiro estágio, a vizinhança é baseada apenas em modos de execução e, para casos em que é possível melhorar o *makespan* encontrado até então, o segundo estágio é executado com vizinhanças baseadas em atividades, solucionando-se o subproblema RCPS. Outra abordagem via SA foi proposta em [Afshar-Nadjafi \(2014\)](#). Porém, o objetivo é minimizar o custo da utilização de recursos. Também foram consideradas datas de *release* para a disponibilidade de recursos.

No trabalho de [Mika et al. \(2008\)](#) foi proposta uma implementação da metaheurística Busca Tabu (BT) para o MRCPSP, considerando tempos de *setup*. Em [Tchao e Martins \(2008\)](#) também foi utilizado um procedimento via BT, combinado com *Path Relinking* como procedimento de intensificação. Já [Atli e Kahraman \(2012\)](#) combinaram uma implementação da BT com várias Regras de prioridade.

[Zhang et al. \(2006\)](#) implementaram a estratégia evolucionária *Particle Swarm Optimization* (PSO), com adição de um procedimento para reparar soluções inviáveis. Também em [Jarboui et al. \(2008\)](#) foi utilizada a metaheurística PSO para atribuir modos de execução às atividades, combinando com uma busca local, que determina o sequenciamento das atividades durante a evolução do PSO. A mesma abordagem via PSO foi implementada por [Truong e Kachitvichyanukul \(2007\)](#). Porém, os autores hibridizaram o PSO com um Algoritmo Genético (AG), sendo que o PSO gera as prioridades das atividades, enquanto o AG define modos de execução para as atividades. [Khalilzadeh et al. \(2012\)](#) também utiliza um PSO, considerando, como objetivo, a minimização do custo de utilização de recursos. [Cui e Yu \(2014\)](#) utilizaram uma abordagem puramente via PSO.

Em [Damak et al. \(2009\)](#), é implementada a abordagem Evolução Diferencial (ED), em que os autores restringem o tempo de busca pelo ED comparando-o com resultados obtidos por [Jarboui et al. \(2008\)](#) e [Bouleimen e Lecocq \(2003\)](#). Já [Cheng](#)

e Tran (2016) hibridizaram um algoritmo ED com o algoritmo de clusterização *C-means*. Zhang et al. (2015) hibridizaram as metaheurísticas PSO e ED, sendo que a primeira resolve o subproblema RCPSP, enquanto a segunda busca pelos melhores modos de execução.

Alcaraz et al. (2003) implementaram um AG, com modificações no cruzamento e mutação postas a partir de observações dos mesmos em seus trabalhos com o RCPSP, assim como modificações observadas na função objetivo do AG proposto por Hartmann (2001). Nesse último trabalho, é apresentada a heurística de busca local *multi mode left shift*, baseada em uma *bounding rule* do algoritmo exato de Sprecher et al. (1997), sendo que Hartmann (2001) hibridizou a busca local com um AG. Em Van Peteghem e Vanhoucke (2010) e Tseng e Chen (2009), também é utilizado um AG, combinado com uma extensão da heurística *Forward/Backward Improvement* - FBI (cf. Li e Willis (1992) e Özdamar e Ulusoy (1996)), sendo que o primeiro trabalho é apontado como o melhor trabalho na literatura para resolução de instâncias em larga escala (cf. Van Peteghem e Vanhoucke (2014)). Em Lova et al. (2009), também é implementado um AG, mas é aplicada uma variação da FBI somente em indivíduos viáveis. Em Bilolikar et al. (2012), é sugerida uma hibridização do AG com um SA. O trabalho de Okada et al. (2014) também faz hibridização do AG com os procedimentos de busca local FBI e CPM. Em Bilolikar (2014) é apresentado um AG com o objetivo de minimizar o custo total do projeto.

Elloumi e Fortemps (2010) implementaram um AG com uma heurística para viabilizar soluções inviáveis em relação a recursos não renováveis. Entretanto, os autores trabalharam no espaço bi-objetivo do problema MRCPSP, considerando a segunda função objetivo a minimização da quantidade utilizada de recursos não renováveis. Como avaliação das soluções, foram utilizadas as abordagens *rank-based* e heurística de clusterização, ambas comuns ao MOGA. Outra abordagem multi-objetivo foi proposta em Kazemi e Tavakkoli-Moghaddam (2011), sendo que os autores utilizaram o NSGA-II (*Nondominated Sorting Genetic Algorithm II*) para resolução de instâncias em larga escala. Uma função a ser otimizada é conhecida como *net present value*, que corresponde a maximização da diferença entre fluxo positivo e negativo de caixa (valor líquido do projeto), já a segunda função maximiza a diferença do *makespan* com a data de entrega do projeto. Nesse último trabalho, não há utilização de recursos não renováveis. Já em Andreica e Chira (2014), foi utilizada outra abordagem via AG com um procedimento para reparar soluções inviáveis em relação a recursos não renováveis e também um operador de cruzamento denominado *Best-Order Crossover*, que utiliza informações do melhor indivíduo na geração corrente, em conjunto com informações dos pais para gerar um novo indivíduo para a próxima geração. O trabalho de Phruksaphanrat (2014) apresenta uma modelagem *fuzzy* para minimizar o *makespan* e o custo total do projeto.

A abordagem *Scatter-search* foi implementada em Van Peteghem e Vanhoucke (2011), que também utiliza um procedimento heurístico para gerar soluções viáveis, assim como três procedimentos para aprimorar uma solução e outros dois procedimentos de busca local. Um procedimento via Colônia de Formigas (ACF) foi utilizado por Chiang et al. (2008) e Li e Zhang (2013), sendo que no último são utilizadas duas regras de prioridade para guiar “formigas” na escolha de atividades na construção do sequenciamento. Uma abordagem via Sistema Imunológico Artificial (SIA) foi utilizada por Van Peteghem e Vanhoucke (2009), hibridizando-a com

um procedimento para controlar as soluções inviáveis na população. Em [Agarwal e Bhat \(2013\)](#), é implementado um algoritmo evolutivo com seleção determinística para os modos de execução.

Em [Muller \(2011\)](#) foi utilizada uma abordagem conhecida como *Adaptive Large Neighborhood Search* (ALNS). Essa metaheurística utiliza de várias estruturas de vizinhança, sendo que, ao longo da busca, as estruturas de vizinhança competem e apenas uma é selecionada para tentativa de melhora da solução corrente. O autor fez uso de limites inferiores e propõe, também, uma extensão da busca local *multi mode left shift*.

Os trabalhos [Wang e Fang \(2012\)](#) e [Soliman e Elgendi \(2014a\)](#) implementam a abordagem *Estimation of Distribution Algorithm* (EDA), que é um algoritmo estocástico populacional baseado em aprendizado estatístico. Diferente de um AG, o EDA produz soluções de acordo com algum modelo probabilístico. Esse modelo probabilístico aplicado ao MRCPSP refere-se à escolha da ordem das atividades, assim como da atribuição de modos, a cada geração, para então produzir um sequenciamento. Logo, os autores criam mecanismos para atribuir critérios de seleção para definir uma ordem em que as atividades deverão ser sequenciadas e, também, definir critérios para atribuição de modos de execução. O EDA tem por definição uma característica de exploração do ambiente de busca e não de intensificação. Em ambos os trabalhos, a busca local FBI é utilizada em conjunto com um procedimento baseado em *random walk*. [Soliman e Elgendi \(2014b\)](#) também implementa um procedimento EDA híbrido com o algoritmo de clusterização *fuzzy c-means*. Nesse trabalho, cada *cluster* é avaliado por sua melhor solução e, então, as melhores soluções em cada *cluster* são selecionadas para uma busca local (FBI e *random walk*).

Abordagens baseadas em multi-agentes, aprendizado e hiper-heurísticas são encontradas nos trabalhos de [Knotts et al. \(2000\)](#), [Jedrzejowicz e Ratajczak \(2006\)](#), [Jedrzejowicz e Ratajczak-Ropel \(2007\)](#), [Wauters et al. \(2011\)](#), [Mirzaei e Akbarzadeh-T \(2013\)](#) e [Messelis e De Causmaecker \(2014\)](#).

A abordagem conhecida como *Shuffled Frog-Leaping Algorithm* (SFLA) foi utilizada em [Wang e Fang \(2011\)](#). De acordo com os autores, esse algoritmo combina a evolução de cada solução, como a proposta de um Algoritmo Memético, e a interação entre as soluções, como proposta do PSO. Os autores combinam o SFLA com a utilização do método *regret-based biased random sampling* e a heurística de busca local FBI. [Khalilzadeh \(2015\)](#) utilizou uma abordagem conhecida como *Honey Bee Swarm Optimization* (HBSO) para o MRCPSP, considerando como objetivo a minimização do custo de utilização de recursos. Os autores também combinaram o HBSO com uma busca local baseada no movimento *Local Left Shift*.

Os trabalhos em [Kolisch e Hartmann \(1999\)](#), [Brucker et al. \(1999\)](#), [Hartmann e Kolisch \(2000\)](#), [Kolisch e Hartmann \(2006\)](#) e [Van Peteghem e Vanhoucke \(2014\)](#) apresentam estudos de comparações entre algumas dessas metaheurísticas utilizadas na resolução do MRCPSP.

A Tabela 3.1 apresenta, resumidamente, algumas das metaheurísticas citadas anteriormente, para resolução do MRCPSP, de acordo com autores, métodos, regras de prioridade (RP), sequenciadores (SGS), uso de busca local (BL) e tipo de recurso considerado.

Tabela 3.1: Heurísticas adotadas na resolução do MRCPSP com minimização do makespan

Autor	Método	RP	SGS	BL	Recursos
Elmaghraby (1977)	CPM	-	-	-	R
Talbot (1982)	B&B truncado	Sim	Serial	-	R/NR
Drexel e Gruenewald (1993)	Multi-pass	Sim	Serial	-	R/NR
Boctor (1993)	Single-pass	Sim	Paralelo	-	R
Özdamar e Ulusoy (1994)	Multi-pass	-	Paralelo	-	R/NR
Kolisch e Drexel (1996)	Multi-pass	Sim	Serial/Paralelo	-	R/NR
Boctor (1996)	Multi-pass	Sim	Paralelo	-	R
Kolisch e Drexel (1997)	Multi-pass	Sim	Serial/Paralelo	Sim	R/NR
Knott et al. (2000)	Multi-agentes	Sim	-	-	R
Jozefowska et al. (2001)	SA	-	Serial	Sim	R/NR
Hartmann (2001)	AG	Sim	Serial	Sim	R/NR
Bouleimen e Lecocq (2003)	SA	-	Serial	Sim	R/NR
Alcaraz et al. (2003)	AG/Multi-pass	Sim	Serial	Sim	R/NR
Zhang et al. (2006)	PSO	-	Serial	Sim	R/NR
Jedrzejowicz e Ratajczak (2006)	Aprendizado	Sim	Serial	Sim	R/NR
Truong e Kachitvichyanukul (2007)	PSO/AG	-	Serial	-	R/NR
Jedrzejowicz e Ratajczak-Ropel (2007)	Multi-agentes	-	Serial	Sim	R/NR
Jarboui et al. (2008)	PSO	-	-	Sim	R/NR
Mika et al. (2008)	BT	-	Serial	Sim	R
Tchao e Martins (2008)	BT/ <i>Path relinking</i>	Sim	-	Sim	R/NR
Chiang et al. (2008)	ACF	-	Serial	-	R/NR
Tseng e Chen (2009)	AG/FBI	-	Serial	Sim	R/NR
Damak et al. (2009)	ED	-	-	-	R/NR
Van Peteghem e Vanhoucke (2009)	SIA	Sim	-	Sim	R/NR
Van Peteghem e Vanhoucke (2010)	AG/FBI	Sim	Serial	Sim	R/NR
Elloumi e Fortemps (2010)	AG	Sim	Serial	Sim	R/NR
Muller (2011)	ALNS	Sim	Serial	Sim	R/NR
Van Peteghem e Vanhoucke (2011)	<i>Scatter-search</i>	Sim	Serial	Sim	R/NR
Wauters et al. (2011)	Multi-agentes	-	Serial	Sim	R/NR
Wang e Fang (2011)	SFLA	Sim	Serial	Sim	R/NR
Khalilzadeh et al. (2012)	PSO	Sim	Paralelo	Sim	R/NR
Bilollikar et al. (2012)	AG/Multi-pass	Sim	Serial	Sim	R/NR
Wang e Fang (2012)	EDA	-	Serial	Sim	R/NR
Atli e Kahraman (2012)	BT	Sim	-	Sim	R/NR
Li e Zhang (2013)	ACF	Sim	Serial	-	R/NR
Agarwal e Bhat (2013)	AE	Sim	-	-	R/NR
Mirzaei e Akbarzadeh-T (2013)	Multi-agentes	-	Serial	-	R/NR
Afshar-Nadjafi (2014)	SA	Sim	Serial	Sim	R/NR
Cui e Yu (2014)	PSO	-	Serial	-	R/NR
Okada et al. (2014)	AG	-	Serial	Sim	R/NR
Andreica e Chira (2014)	AG	-	Serial	-	R/NR
Messelis e De Causmaecker (2014)	Hiper-heurísticas	-	Serial/Paralelo	Sim	R/NR
Soliman e Elgendi (2014a)	EDA	-	Serial	Sim	R/NR
Soliman e Elgendi (2014b)	EDA	-	Serial	Sim	R/NR
Zhang et al. (2015)	PSO/ED	Sim	Serial	-	R/NR
Cheng e Tran (2016)	ED/C-Means	-	Serial	-	R/NR

Alguns dos trabalhos citados não estão presentes na Tabela 3.1, pois alguns desses trabalhos abordaram a resolução de variações do MRCPSP, como objetivo, atividades ou múltiplos projetos.

Pela Tabela 3.1, é verificado que a adoção de algoritmos populacionais para a solução do MRCPSP é usual na literatura, assim como a utilização de técnicas de busca local, regras de prioridade e do SGS Serial. Os algoritmos populacionais recebem destaque, pois apresentam boa capacidade na resolução do MRCPSP. As regras de prioridade agregam características inerentes do MRCPSP às metaheurísticas. Conforme discutido em Kolisch (1996b), existe diferença no conjunto de soluções gerado

pelo SGS Serial em relação ao SGS Paralelo, sendo que o SGS paralelo nem sempre produz a solução ótima, ao contrário do SGS Serial. Já a utilização de busca local aprimora as soluções construídas pelas metaheurísticas. Regras de prioridade, sequenciadores (SGS) e busca local, comuns ao MRCPSP, são discutidos no capítulo em seguida.

Capítulo 4

Modelos e Algoritmos

Neste Capítulo são apresentadas heurísticas relacionadas ao MRCPSP, assim como as representações de um projeto e solução computacional. As heurísticas discutidas são comuns na literatura para a resolução do MRCPSP. Também são apresentadas as metaheurísticas propostas nesta dissertação para a resolução desse problema.

4.1 Representação de Projetos

Para representar um projeto, é comum na literatura o uso da estrutura em rede *Activity on Node* (AoN) (cf. [Demeulemeester \(2002\)](#)). Nessa estrutura, as atividades são os nós e os arcos denotam as precedências entre atividades, sendo que as atividades estão em uma ordem topológica. A Figura 4.1 apresenta um exemplo dessa estrutura em rede, para um projeto com 12 atividades, sendo 2 fictícias ($j = 1$ e $j = 12$). Já a Tabela 4.1 apresenta os modos de execução das atividades, com seus respectivos tempos de processamento e consumo de recursos, para $K = 4$ tipos de recursos, sendo $k = 1, 2$ representando os recursos renováveis e $k = 3, 4$ representando os recursos não renováveis.

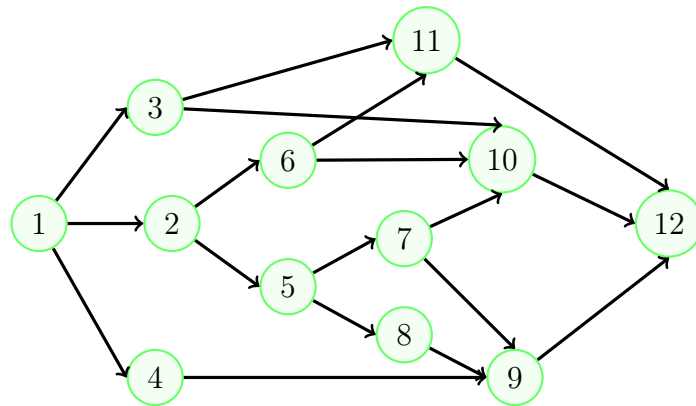


Figura 4.1: Exemplo de projeto em estrutura de rede AoN. Adaptação da Instância j102_2.mm da PSPLIB (cf. [Kolisch e Sprecher \(1997\)](#))

Tabela 4.1: Modos de execução para as atividades no exemplo de projeto apresentado pela Figura 4.1. Adaptação da Instância j102_2.mm da PSPLIB.

j	m	d_{jm}	r_{jmk}^ρ	r_{jmk}^ν	$R_k^\rho, k = 1, 2$	$R_k^\nu, k = 3, 4$
1	1	0	{0,0}	{0,0}	{9,4}	{29,40}
2	1	3	{6,0}	{9,0}		
2	2	9	{5,0}	{0,8}		
3	1	1	{0,4}	{0,8}		
3	2	1	{7,0}	{0,8}		
3	3	5	{0,4}	{0,5}		
4	1	5	{7,0}	{2,0}		
4	2	8	{6,0}	{0,7}		
5	1	6	{2,0}	{0,7}		
6	1	2	{2,0}	{8,0}		
6	2	6	{2,0}	{0,1}		
7	1	3	{5,0}	{10,0}		
7	2	8	{5,0}	{0,10}		
8	1	4	{6,0}	{0,1}		
8	2	10	{3,0}	{10,0}		
8	3	10	{4,0}	{0,1}		
9	1	2	{2,0}	{6,0}		
9	2	7	{1,0}	{0,8}		
9	3	10	{1,0}	{0,7}		
10	1	1	{4,0}	{4,0}		
10	2	1	{0,2}	{0,8}		
10	3	9	{4,0}	{0,5}		
11	1	6	{0,2}	{0,10}		
11	2	9	{0,1}	{0,9}		
11	3	10	{0,1}	{0,7}		
12	1	0	{0,0}	{0,0}		

4.2 Método do Caminho Crítico

O Método do Caminho Crítico (*Critical Path Method* - CPM), datado dos anos 50, e discutido em Kelley Jr e Walker (1959), identifica atividades críticas em um projeto, além de fornecer um limite inferior para a duração total do projeto. O CPM é responsável por identificar os tempos EST_j , LST_j , EFT_j e LFT_j de uma atividade j , por meio de um procedimento denominado *Forward and backward pass*, que desconsidera as restrições de recursos. Os valores $[EST_j, EFT_j]$ definem uma janela em que a atividade j pode começar e terminar o mais cedo possível, sem violação da restrição de precedência. Já os valores $[LST_j, LFT_j]$ definem a janela em que a atividade j pode começar e terminar o mais tarde possível, sem violação da restrição de precedência e atraso no projeto. A janela temporal $[EST_j, LFT_j]$ define os períodos possíveis em que uma atividade deve começar e terminar, sem causar atrasos no projeto. Assumindo uma representação AoN, o Algoritmo 1 descreve o CPM.

Algoritmo 1: Método do Caminho Crítico

```

1 Inicializar  $EST_1 \leftarrow EFT_1 \leftarrow 0$  ;
2 para  $j \leftarrow 2$  até  $|J|$  faça
3    $EST_j \leftarrow \max\{EFT_i : i \in \mathcal{P}_j\}$  ;
4    $EFT_j \leftarrow EST_j + d_j$  ;
5 fim
6 Inicializar  $LFT_{|J|} \leftarrow LST_{|J|} \leftarrow EFT_{|J|}$  ;
7 para  $j \leftarrow |J| - 1$  até 1 faça
8    $LFT_j \leftarrow \min\{LST_i : i \in \mathcal{S}_j\}$  ;
9    $LST_j \leftarrow LFT_j - d_j$  ;
10 fim

```

Nota-se, pelo Algoritmo 1, que a definição da janela $[EST_j, LFT_j]$ é calculada somente identificando as precedências entre as atividades e não se verifica a restrição de recursos. Segundo Demeulemeester (2002), a complexidade do CPM é da ordem $O(n^2)$, sendo n o número de atividades.

Após calcular os tempos EST_j , LST_j , EFT_j e LFT_j de cada atividade j , a folga de uma atividade SLK_j é definida como:

$$SLK_j = LST_j - EST_j = LFT_j - EFT_j \quad (4.0)$$

As atividades que possuem folga igual a zero são chamadas de atividades críticas (Kelley Jr e Walker, 1959). Esse cálculo equivale a encontrar o maior caminho no grafo representado pela AoN. Como exposto em Demeulemeester (2002), podem existir vários caminhos críticos em um projeto. Para o MRCPSP, atribuindo a cada atividade o modo com menor duração e executando o CPM, obtém-se o Caminho Crítico Minimal. Por outro lado, atribuindo a cada atividade o modo com maior duração, obtém-se o Caminho Crítico Maximal.

Como ilustração da execução do Algoritmo 1, a Tabela 4.2 apresenta uma possível atribuição de modo de execução para cada atividade, assim como os valores calculados para as variáveis EST, LST, EFT, LFT e SLK.

Tabela 4.2: Exemplo de execução do Algoritmo 1 sobre o problema de projeto apresentado pela Figura 4.1

j	m	d_{jm}	EST	EFT	LST	LFT	SLK
1	1	0	0	0	0	0	0
2	1	3	0	3	0	3	0
3	1	1	0	1	8	9	8
4	2	8	0	8	9	17	9
5	1	6	3	9	7	13	4
6	2	6	3	9	3	9	0
7	1	3	9	12	14	17	5
8	1	4	9	13	13	17	4
9	1	2	13	15	17	19	4
10	2	1	12	13	18	19	6
11	3	10	9	19	9	19	0
12	1	0	19	19	19	19	0

A Tabela 4.2 informa, para a atribuição de modos definida, que o valor do caminho crítico é de 19 períodos de tempo. Além do mais, as atividades pertencentes ao caminho crítico são 1, 2, 6, 11 e 12, pois apresentam SLK igual a zero. Outra informação útil presente nessa tabela é o valor final do *makespan*. Se todas as atividades estiverem dispostas no tempo, dentro da janela $[EST, LFT]$, satisfazendo a disponibilidade de recursos renováveis ao longo dos 19 períodos de tempo, então o projeto, certamente, será finalizado com *makespan* igual a 19 unidades de tempo. Caso contrário, possivelmente, o projeto sofrerá atraso.

4.3 Representação das Soluções

Para realizar buscas no espaço de solução de um problema de otimização, é necessário definir como representar computacionalmente uma solução. Várias estruturas de representação de soluções para o MRCPSP podem ser encontradas na literatura, como as apresentadas em Demeulemeester (2002) e em Kolisch e Hartmann (1999). No presente trabalho, uma solução s é representada por um par de listas $s = (L_J, L_M)$, em que L_J é uma permutação viável de todas as atividades e L_M uma permutação válida de modos de execução. Assim, $L_J[i] = j$ e $L_M[i] = m_j$ informam que, na posição i das listas L_J e L_M , a atividade j é executada pelo modo m .

A lista L_J deve ser viável em relação à restrição de precedência entre as atividades. Logo, para uma atividade j na posição i da lista de atividades, todos os seus predecessores $p_j \in \mathcal{P}_j$, e sucessores $s_j \in \mathcal{S}_j$, estão, respectivamente, em posições i_p e i_s , tal que $i_p < i < i_s$.

A Figura 4.2 apresenta um exemplo de solução desse esquema de representação, a partir do projeto apresentado pela Figura 4.1.

L_J	1	2	5	7	6	3	8	4	11	10	9	12
L_M	1	1	1	1	2	1	1	2	3	2	1	1

Figura 4.2: Exemplo de solução para o problema de projeto representado pela Figura 4.1.

Para construir a lista de atividades apresentada pela Figura 4.2, a atividade 1 é a primeira a entrar na lista. Então, as atividades 2, 3 e 4 estão disponíveis, pois todos os seus predecessores já foram selecionados para a lista. Por algum mecanismo de escolha, determinou-se que a atividade 2 será a próxima a entrar para a lista de atividades. Em seguida, estão disponíveis as atividades 3, 4, 5 e 6. O mesmo critério de escolha determinou que a atividade 5 deverá ser a próxima para entrar na lista. Por fim, constrói-se a solução apresentada pela Figura 4.2. É importante destacar que as atividades 1 e 12 devem ser, respectivamente, a primeira e a última atividades da lista de atividades. O mecanismo que determina a ordem de escolha entre as atividades será abordado na Seção 4.5 no item de seção Regras de Prioridade.

4.4 Esquemas de Geração de Sequenciamento

A partir de uma atribuição de modos de execução para cada atividade em um projeto, o próximo passo é, então, sequenciá-las, respeitando as restrições de precedência e consumo de recursos. Esquemas de Geração de Sequenciamento (*Schedules Generation Schemes - SGS*) realizam essa tarefa. Para sequenciar um conjunto de atividades, é necessário sequenciar uma atividade inicial e, em seguida, todas as suas sucessoras em algum instante após a atividade inicial. Esse processo segue até que a última atividade seja sequenciada.

Dois tipos de esquemas de sequenciamento são conhecidos - métodos Serial e Paralelo. Ambos são descritos em Kolisch (1996b). Esses métodos geram sequenciamentos viáveis a partir da extensão de um sequenciamento parcial (i.e. um sequenciamento realizado em um subconjunto das atividades). Para isso, um conjunto de atividades elegíveis para o sequenciamento é construído, chamado de conjunto de decisão D_n . Quando uma atividade é elegível, ou seja, está em D_n , significa que todos os seus predecessores já foram sequenciados. Estes métodos são descritos a seguir.

4.4.1 Método Serial

O Esquema de Geração de Sequenciamento Serial (*Serial Scheduling Scheme - SSS*), proposto por Kelley (1963), consiste em, dado um conjunto de $n = 1, \dots, |J|$ estágios, a cada estágio uma atividade é selecionada e sequenciada. Também são definidos dois conjuntos dinâmicos de atividades, nomeados como S_n e D_n . O conjunto S_n descreve as atividades já sequenciadas ou, ou seja, o sequenciamento parcial. Já o conjunto D_n é o conjunto de decisão, que contém as atividades ainda não sequenciadas, dado que cada predecessor $p_j \in \mathcal{P}_j$ de uma atividade $j \in D_n$ está em S_n , ou seja:

$$D_n = \{j : j \notin S_n, \mathcal{P}_j \subseteq S_n\}$$

Algoritmo 2: Esquema de Geração de Sequenciamento Serial - (SSS)

```

1 Inicializar  $n \leftarrow 1, S_n \leftarrow \emptyset$  ;
2 enquanto  $|S_n| < |J|$  faça
3   Calcular  $D_n$  e  $\pi R_{kt}, t = 1, \dots, T, k = 1, \dots, K^\rho$  ;
4    $j^* \leftarrow \min_{j \in D_n} \{j : v(j) = \min_{i \in D_n} \{v(i)\}\}$  ;
5    $EFT_{j^*} \leftarrow \max\{FT_i : i \in \mathcal{P}_{j^*}\} + d_{j^*m}$  ;
6    $FT_{j^*} \leftarrow \min\{t : EFT_{j^*} \leq t \leq T, r_{j^*mk} \leq \pi R_{k\tau}, \tau = t - d_{j^*m}, \dots, t, k = 1, \dots, K^\rho\}$  ;
7    $S_{n+1} \leftarrow S_n \cup \{j^*\}$  ;
8    $n \leftarrow n + 1$  ;
9 fim

```

A cada estágio n , uma atividade $j \in D_n$ é selecionada por meio de um critério de seleção, removida de D_n e, após, acrescentada em S_n , de modo que a atividade j foi sequenciada. O método finaliza no estágio $n = |J|$, quando todas as atividades do projeto estiverem em S_n .

Seja, então, πR_{kt} definido como o restante disponível do recurso renovável k no período t , dado por:

$$\pi R_{kt} = R_k^\rho - \sum_{j \in S_t} r_{jmk}^\rho$$

Além disso, seja T um limite superior para a duração do projeto e $v(j)$ um valor associado a $j \in D_n$. Então, [Kolisch \(1996b\)](#) apresenta o SSS de acordo com o Algoritmo 2.

Pela linha 4 no Algoritmo 2, uma atividade j^* será escolhida como aquela que apresentar o menor valor de $v(j), \forall j \in D_n$, com o menor índice como quebra de empate. Já a linha 5 determina o tempo de término mais cedo para j^* , baseado no maior tempo de término entre todos os seus predecessores. A linha 6 apresenta a atribuição do menor tempo de término viável em que j^* será finalizada, respeitando o consumo de recursos durante todo o período em que j^* é executada. Após o final da repetição dada pelas linhas 3-8, todas as atividades receberam um tempo de término, ou seja, foram sequenciadas. A complexidade do SSS é da ordem de $O(n^2k)$, segundo [Demeulemeester \(2002\)](#). A Tabela 4.3 apresenta o passo-a-passo da execução do SSS por meio da atribuição de modos apresentada pela Figura 4.2. Nessa tabela, o símbolo “-” (corte) indica que uma atividade j^* foi removida de D_n . Já a Figura 4.3 mostra o sequenciamento obtido ao término do SSS.

Tabela 4.3: Exemplo de execução passo-a-passo do Algoritmo SSS.

n	D_n	j^*	EFT_{j^*}	FT_{j^*}	S_n
1	{1}	1	0	0	{1}
2	{2,3,4}	2	3	3	{1,2}
3	{3,4,5,6}	5	9	9	{1,2,5}
4	{3,4,6,7,8}	7	12	12	{1,2,5,7}
5	{3,4,6,8}	6	9	9	{1,2,5,7,6}
6	{3,4,8}	3	1	1	{1,2,5,7,6,3}
7	{4,8,10,11}	8	13	16	{1,2,5,7,6,3,8}
8	{4,10,11}	4	8	24	{1,2,5,7,6,3,8,4}
9	{10,11,9}	11	19	19	{1,2,5,7,6,3,8,4,11}
10	{10,9}	10	13	13	{1,2,5,7,6,3,8,4,11,10}
11	{9}	9	26	26	{1,2,5,7,6,3,8,4,11,10,9}
12	{12}	12	26	26	{1,2,5,7,6,3,8,4,11,10,9,12}

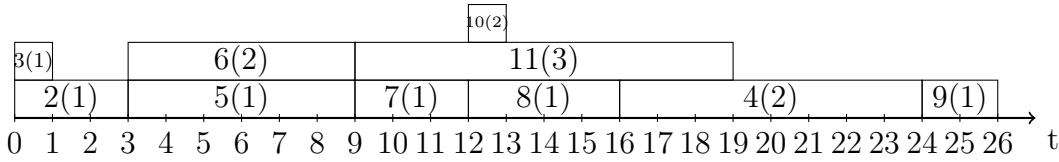


Figura 4.3: Exemplo de sequenciamento obtido pelo SSS a partir da representação dada pela Figura 4.2

Pela Tabela 4.3, percebe-se que, a cada estágio n , somente uma atividade é sequenciada (colocada em S_n) e que a ordem da escolha das atividades, a cada estágio, está de acordo com a lista L_J apresentada pela Figura 4.2. No sequenciamento dado pela Figura 4.3, cada atividade é representada por $j(m)$, tal que j é a atividade e m o modo de execução atribuído. O sequenciamento é viável em relação à disponibilidade dos recursos não renováveis, $R_3^\nu = 29$ e $R_4^\nu = 40$, pois a quantidade requerida pelas atividades, para os modos atribuídos, é tal que $\sum_{j=1}^{12} r_{jm3}^\nu = 25$ e $\sum_{j=1}^{12} r_{jm4}^\nu = 39$. O *makespan* do projeto é de 26 unidades de tempo. A disponibilidade dos recursos renováveis, $R_1^\rho = 9$ e $R_2^\rho = 4$, é respeitada ao longo de todo o horizonte de realização do projeto. Nota-se também que as atividades fictícias 1 e 12 não aparecem no sequenciamento, pois apresentam duração nula e não consomem quaisquer recursos. Verificando a Tabela 4.2, percebe-se que as atividades críticas 2, 6 e 11 estão em sua janela $[EST, LFT]$ calculada respectivamente, como $[0, 3]$, $[3, 9]$ e $[9, 19]$. Isso acontece com as outras atividades, exceto com as atividades 4 e 9, cujas janelas $[EST, LFT]$ são, respectivamente, $[0, 17]$ e $[13, 19]$. Esse atraso nas atividades acontece pela indisponibilidade de recursos renováveis, dentro dessas janelas de tempo, devido à concorrência com outras atividades nesses mesmos períodos de tempo. Isso criou um atraso no projeto, sendo o mesmo finalizado com *makespan* maior que o caminho crítico de 19 unidades de tempo.

4.4.2 O método Paralelo

De acordo com Kolisch (1996b), existem duas associações ao esquema de sequenciamento em paralelo (*Parallel Scheduling Scheme - PSS*), que são os algoritmos de Kelley (1963) e Bedworth e Bailey (1982). O esquema em paralelo consiste no máximo de $|J|$ estágios, sendo que, a cada estágio, um conjunto de atividades é sequenciada. Como descrito em Kolisch (1996b), cada estágio n é associado a um tempo de sequenciamento t_n . Devido a esse tempo, o conjunto de atividades sequenciadas é dividido nos subconjuntos: Conjunto completo C_n , que representa atividades que foram sequenciadas e completadas até o tempo de sequenciamento, enquanto o conjunto ativo A_n representa atividades que foram sequenciadas, entretanto não completadas, ou seja, estão ativas no tempo de sequenciamento corrente. Finalmente, o já conhecido conjunto de decisão D_n , que de acordo com Kolisch (1996b), em contraste com o método serial, D_n contém todas as atividades não sequenciadas que estão disponíveis para o sequenciamento em relação as restrições de precedência e consumo de recursos.

Antes da apresentação do algoritmo PSS, as seguintes definições são necessárias. Dados os conjuntos A_n e C_n , então seja πR_{kt_n} o restante disponível do recurso k no tempo de sequenciamento t_n , e o conjunto D_n , definidos, respectivamente, como:

$$\pi R_{kt_n} = R_k^\rho - \sum_{j \in A_n} r_{jmk}^\rho$$

$$D_n = \{j : j \notin \{C_n \cup A_n\}, \mathcal{P}_j \subseteq C_n, r_{jmk}^\rho \leq \pi R_{kt_n}, k = 1, \dots, K^\rho\}$$

Então, Kolisch (1996b) apresenta a descrição formal do método PSS de acordo com o Algoritmo 3.

Algoritmo 3: Esquema de Geração de Sequenciamento Paralelo - (PSS)

```

1 Inicializar
   $n \leftarrow 1, t_n \leftarrow 0, D_n \leftarrow \{1\}, A_n \leftarrow C_n \leftarrow \emptyset, \pi R_{kt_n} \leftarrow R_k^\rho, k = 1, \dots, K^\rho,$ 
  GOTO Passo (ii) ;
2 enquanto  $|A_n \cup C_n| < |J|$  faça
3   Passo (i) ;
4    $t_n \leftarrow \min\{FT_j : j \in A_{n-1}\}$  ;
5    $A_n \leftarrow A_{n-1} \setminus \{j : j \in A_{n-1}, FT_j = t_n\}$  ;
6    $C_n \leftarrow C_{n-1} \cup \{j : j \in A_{n-1}, FT_j = t_n\}$  ;
7   Calcular  $\pi R_{kt_n}, k = 1, \dots, K^\rho$  e  $D_n$  ;
8   Passo (ii) ;
9    $j^* \leftarrow \min_{j \in D_n} \{j : v(j) = \min_{i \in D_n} \{v(i)\}\}$  ;
10   $FT_{j^*} \leftarrow t_n + d_{j^*m}$  ;
11   $A_n \leftarrow A_n \cup \{j^*\}$  ;
12  Calcular  $\pi R_{kt_n}, k = 1, \dots, K^\rho$  e  $D_n$  ;
13  se  $D_n \neq \emptyset$  então
14    GOTO Passo (ii) ;
15  senão
16     $n \leftarrow n + 1$  ;
17  fim
18 fim
```

Pelo Algoritmo 3, cada estágio é composto de dois passos: (i) O novo tempo de sequenciamento t_n é determinado e atividades com tempo de término igual a t_n são removidas de A_n e colocadas em C_n . Com isso, possivelmente, algumas atividades serão acrescentadas em D_n ; (ii) Uma atividade $j \in D_n$ é selecionada pelo valor $v(j)$ e sequenciada com início em t_n . Em seguida, essa atividade é removida de D_n e colocada em A_n .

Percebe-se que o passo (ii) é executado até que o conjunto de decisão esteja vazio, ou seja, atividades foram sequenciadas ou não estão mais disponíveis para sequenciamento em relação à restrição de recursos. O PSS finaliza sua execução quando todas as atividades estão em C_n ou A_n . A complexidade do PSS também é da ordem de $O(n^2k)$ (Demeulemeester, 2002).

A Tabela 4.4 apresenta o passo-a-passo da execução do Algoritmo 3 por meio da atribuição de modos apresentada pela Figura 4.2.

Tabela 4.4: Exemplo de execução passo-a-passo do Algoritmo PSS

n	t	D_n	j^*	FT_{j^*}	A_n	C_n
1	0	{1}	1	0	{1}	{}
2	0	{2,3,4}	2	3	{2}	{1}
2	0	{3}	3	1	{2,3}	{1}
3	1	{}			{2}	{1,3}
4	3	{4,5,6}	5	9	{5}	{1,3,2}
4	3	{4,6}	6	9	{5,6}	{1,3,2}
4	3	{}			{5,6}	{1,3,2}
5	9	{4,7,8,11}	7	12	{7}	{1,3,2,5,6}
5	9	{11}	11	19	{7,11}	{1,3,2,5,6}
6	12	{4,8,10}	8	16	{11,8}	{1,3,2,5,6,7}
6	12	{10}	10	13	{11,8,10}	{1,3,2,5,6,7}
7	13	{}			{11,8}	{1,3,2,5,6,7,10}
8	16	{4}	4	24	{11,4}	{1,3,2,5,6,7,10,8}
9	19	{}			{4}	{1,3,2,5,6,7,10,8,11}
10	24	{9}	9	26	{9}	{1,3,2,5,6,7,10,8,11,4}
11	26	{12}	12	26	{12}	{1,3,2,5,6,7,10,8,11,4,9}

Pela Tabela 4.4, percebe-se que, a cada estágio n , mais de uma atividade pode ser sequenciada (colocada em A_n) se, e somente se, houver recursos disponíveis no tempo de sequenciamento t_n . A ordem da escolha das atividades, quando estão no conjunto D_n , a cada estágio, está de acordo com a lista L_J apresentada pela Figura 4.2. Percebem-se, também, as atualizações em D_n a cada estágio. Por exemplo, no início do estágio $n = 4$, $t_n = 3$, $C_n = \{1, 3, 2\}$ e $D_n = \{4, 5, 6\}$. Ao selecionar a atividade 5, exclui-se 5 de D_n e a acrescenta em A_n . No segundo momento, seleciona-se a atividade 6, porém a atividade 4 torna-se inviável para o tempo $t_n = 3$ devido a indisponibilidade de recursos nesse período. Então, 4 também é removida de D_n .

Após a execução do PSS, o mesmo sequenciamento representado pela Figura 4.3 é gerado. Como discutido em Sprecher et al. (1995) e Kolisch (1996b), nem sempre os sequenciamentos obtidos pelos métodos SSS e PSS serão iguais. O conjunto de soluções X_{sss} produzido pelo SSS contém a solução ótima, ao contrário do conjunto

de soluções X_{pss} , produzido pelo PSS, que nem sempre contém a solução ótima. Logo, a relação de cardinalidade entre os conjuntos é tal que $|X_{sss}| \geq |X_{pss}|$ (Sprecher et al., 1995). Contudo, ambos os métodos são comuns em procedimentos heurísticos encontrados na literatura.

É importante esclarecer que, ao utilizar uma lista de atividades L_J , tal lista já determina a ordem em que as atividades podem ser selecionadas, de acordo com a linha 4 do SSS e linha 9 do PSS. Ou seja, é comum na literatura construir a lista de atividades pelos valores $v(j)$, $\forall j \in J$, e, então, sequenciar as atividades, de acordo com a ordem em que aparecem na lista sem a necessidade de executar as linhas 4 do SSS e 9 do PSS.

Os trabalhos em Kolisch e Drexl (1996), Kolisch (1996b) e Lova et al. (2006) apresentam um estudo estatístico entre os algoritmos 2 e 3, concluindo que há diferenças nos sequenciamentos calculados por estes métodos, quando submetidos a certas instâncias e combinados com outras heurísticas, como Regras de Prioridade.

4.5 Heurísticas Construtivas

4.5.1 Regras de Prioridade

Tanto o SSS quanto o PSS necessitam retirar uma atividade do conjunto elegível D_n para determinar um tempo de início para essa atividade. Ou até mesmo, como comentado, a lista de atividades é construída de acordo com algum critério. Há, também, a necessidade de atribuir um modo de execução para cada atividade. Em ambos os casos utiliza-se uma Regra de Prioridade.

Regras de prioridade definem valores de prioridade $v(j)$, para uma atividade j e para modos de execução m_j , permitindo atribuir modos de execução a atividades e permitindo aos métodos PSS ou SSS sequenciarem a melhor atividade j^* , em algum instante de tempo, baseando-se no seu valor de prioridade. A escolha da melhor atividade j^* se faz em conjunto com um objetivo (min ou max), e, assim, é construído um sequenciamento. Logo, regras de prioridade são heurísticas que geram vários sequenciamentos (Kolisch, 1996b).

Em relação ao MRCPSP, tanto para modos de execução quanto para atividades, as regras de prioridade estão relacionadas a tempo e/ou recursos e serão discutidas em seguida.

4.5.1.1 Regras de prioridade para modos

Um modo de execução viável é tal que a quantidade requerida de recursos não ultrapassa a quantidade disponível para todo o horizonte do projeto (recursos não renováveis) ou para algum período em específico (recursos renováveis) (Boctor, 1993). Exemplos de regras de prioridade apresentadas pela literatura, para os modos de execução, são mostradas na Tabela 4.5 e discutidas em seguida.

Tabela 4.5: Exemplos de regras de prioridade para modos de execução

Regra de Prioridade	Objetivo	$v(j)$
<i>Shortest Feasible Mode</i> - SFM	min	d_{jm}
<i>Least Resource Proportion</i> - LRP	min	$\max_k \left\{ \frac{r_{mk}}{R_k} \right\}$
<i>Stochastic Scheduling Method</i> - STOCOM1	max	ω_{jm}
<i>Minimum Resource Demand Mode</i> - Max-RDM	min	$\sum_{k=1}^{K^\rho} r_{jmk}^\rho d_{jm}$
<i>Maximum Resource Demand Mode</i> - Min-RDM	max	$\sum_{k=1}^{K^\rho} r_{jmk}^\rho d_{jm}$
<i>Minimum Normalized Resources</i> - MNR	min	$\sum_{k=1}^{K^\nu} \frac{r_{jmk}^\nu}{R_k^\nu}$
<i>Random</i> - RAND	-	$1/ M_j $

O trabalho de Boctor (1993) foi o primeiro a apresentar regras de prioridade baseadas em modos de execução para a resolução do MRCPPSP, assim como foi o primeiro trabalho a avaliar, por experimentos computacionais e estatísticos, estas regras. Boctor (1993) avaliou as regras de prioridade para modos de execução SFM, LRP e uma outra regra conhecida como *Least Critical Resource* (LCR). A regra SFM escolhe o modo viável, em relação aos recursos não renováveis, com menor duração. Como discutido em Boctor (1993), executando as atividades no modo que fornece a menor duração parece ser a melhor escolha, se o objetivo é minimizar o tempo de duração do projeto. Entretanto, pode haver o efeito contrário. Os modos com menor duração são aqueles que consomem uma maior quantidade de recursos, e, neste caso, a regra SFM força um atraso das atividades finais que poderiam ser executadas em paralelo com atividades iniciais. Consequentemente, a duração do projeto pode ser prolongada. Já as regras LCR e LRP, de acordo com Boctor (1993), buscam sequenciar o máximo de atividades em paralelo, por meio da disponibilidade de recursos.

Por meio dos experimentos, Boctor (1993) concluiu que a regra SFM apresentou melhores resultados, enquanto a regra LRP apresentou resultados menos satisfatórios.

Drexel e Gruenewald (1993) apresentam uma regra para a atribuição de modos definida como STOCOM1. Esta regra avalia cada par modo-atividade, em relação a duração, somente para todos os modos das atividades elegíveis. Segundo Drexel e Gruenewald (1993), essa regra tem uma tendência de atribuir o modo com menor duração para atividades elegíveis. Kolisch (1995) apresenta uma nova regra, alterando a regra LRP, de maneira que atribui o modo com o menor consumo relativo em relação aos recursos não renováveis. Assim, a regra LRP passa a ser considerada somente para recursos não renováveis. Em Özdamar (1999) é utilizada somente a regra SFM para modos de execução em conjunto com outras regras para atividades. Buddhakulsomsiri e Kim (2007) utilizaram as regras SFM e LRP, porém adaptaram a última para que fosse considerada a duração de cada atividade. Os autores também apresentam testes computacionais nos quais demonstram que a regra SFM apresenta melhores resultados.

Em Van Peteghem e Vanhoucke (2009), uma população de soluções com atribuição aleatória de modos para cada atividade é gerada, porém, os autores utilizam *profit values* para eliminar uma parte da população. Uma abordagem semelhante foi usada em Van Peteghem e Vanhoucke (2011).

No trabalho de [Atli e Kahraman \(2012\)](#), a regra SFM foi adaptada de duas maneiras: (i) *Minimum duration mode*, que seleciona o modo com menor duração; e (ii) *Maximum Duration Mode*, que seleciona o modo com maior duração. Os autores também utilizaram as regras Max-RDM e Min-RDM. Todas as regras foram utilizadas na construção de uma solução inicial.

Os autores [Agarwal e Bhat \(2013\)](#) apresentam uma regra determinística, que atribui um *score* para selecionar o melhor modo de execução a cada iteração do algoritmo evolucionário proposto.

No trabalho de [Colak et al. \(2013\)](#) são utilizadas duas variações da regra SFM. A primeira variação, denominada *Shortest Feasible Mode with Conditional Wait for the Fastest Mode* (SFM-CWFM) define que, se o modo de execução viável mais rápido (ou seja, que apresenta a menor duração) não é o modo mais rápido, é, então, verificado o período em que a atividade deve esperar pelos recursos para ser executada no modo mais rápido e a diferença entre as durações do modo mais rápido e do modo viável mais rápido. Assim, se, para executar a atividade no modo mais rápido, for necessário esperar menor tempo que a diferença entre as durações, faz sentido esperar. Já na segunda variação, denominada *Shortest Feasible Mode with Conditional Wait for a Better Mode* (SFM-CWBM), para uma atividade, serão considerados todos os modos mais rápidos que o modo viável mais rápido. Em seguida, é verificada a diferença entre as durações dos modos mais rápidos e o modo viável mais rápido, assim como o tempo de espera para a execução de cada modo mais rápido que o modo viável mais rápido. A regra SFM-CWBM é uma extensão da regra SFM-CWFM, sendo que a primeira possibilita não só a execução do modo mais rápido, mas também do segundo modo mais rápido em diante, ao contrário da regra SFM-CWFM já discutida. Nesse último trabalho citado, a viabilidade dos modos é verificada somente para os recursos renováveis, pois não foram tratados recursos não renováveis.

[Lova et al. \(2006\)](#) realizaram um estudo comparando regras de prioridade para atividades, utilizando somente a regra SFM para atribuição de modos de execução. Nos trabalhos [Knotts et al. \(2000\)](#), [Jozefowska et al. \(2001\)](#), [Bouleimen e Lecocq \(2003\)](#), [Najid e Arroub \(2010\)](#), [Wauters et al. \(2011\)](#), [Li e Zhang \(2013\)](#), [Soliman e Elgendi \(2014a\)](#), [Bilolika \(2014\)](#) foi utilizada somente a regra SFM na atribuição de modos de execução na construção de uma solução inicial.

Em [Jarboui et al. \(2008\)](#), para cada modo de execução, existe uma probabilidade de escolha dos modos, baseando-se na regra Max-RDM. [Lova et al. \(2009\)](#) também criaram uma probabilidade de escolha dos modos associada à regra MNR, porém, somente para recursos não renováveis.

Vários autores, como [Mori e Tseng \(1997\)](#), [Hartmann \(2001\)](#), [Alcaraz et al. \(2003\)](#), [Zhang et al. \(2006\)](#), [Lorenzoni et al. \(2006\)](#), [Truong e Kachitvichyanukul \(2007\)](#), [Mika et al. \(2008\)](#), [Van Peteghem e Vanhoucke \(2008\)](#), [Tseng e Chen \(2009\)](#), [Damak et al. \(2009\)](#), [Elloumi e Fortemps \(2010\)](#), [Van Peteghem e Vanhoucke \(2010\)](#), [Nader Abadi et al. \(2011\)](#), [Wang e Fang \(2011\)](#), [Wang e Fang \(2012\)](#), [Khalilzadeh et al. \(2012\)](#), [Bilolika et al. \(2012\)](#), [Okada et al. \(2014\)](#), [Afshar-Nadjafi \(2014\)](#), [Cui e Yu \(2014\)](#), [Andreica e Chira \(2014\)](#), [Soliman e Elgendi \(2014b\)](#), [?, Khalilzadeh \(2015\)](#) descrevem a atribuição de modos de forma aleatória na construção de soluções iniciais.

Como pode ser percebido, a literatura destaca a atribuição de modos de execução

para atividades que levam em consideração a duração das atividades. Nota-se, também, destaque para a regra SFM, embora vários autores tenham utilizado a regra RAND e obtiveram resultados relevantes, de acordo com a literatura. Em alguns trabalhos, as regras de prioridade foram utilizadas de forma determinística, já em outros trabalhos os autores criaram um mecanismo de probabilidade de seleção proporcional ao valor de prioridade, ou seja, quanto melhor o valor de prioridade, maior será a possibilidade de seleção de um determinado modo de execução. Esse mecanismo de probabilidade de seleção é detalhado na subseção 4.5.2.

4.5.1.2 Regras de prioridade para atividades

De acordo com Kolisch (1996b), uma heurística para sequenciamento baseada em regras de prioridade é construída por dois componentes: um SGS (*Scheduling Generation Scheme*) e uma regra de prioridade. O SGS é responsável por atribuir tempos de início para as atividades e uma regra de prioridade específica é utilizada, com o objetivo de escolher uma ou mais atividades para o sequenciamento em construção.

Regras de prioridade para atividades, no contexto de sequenciamento em projetos, foram aplicadas ao problema RCPSP e adaptadas ao MRCPPSP. Essas regras de prioridade, como explicado em Demeulemeester (2002), são classificadas em cinco categorias, baseadas no tipo de informação que é requerida para a construção da lista de atividades ou da escolha da atividade na construção de um sequenciamento parcial:

- (i) regras baseadas em atividades: considera informações relacionadas somente às atividades, desconsiderando o restante do projeto. A Tabela 4.6 apresenta algumas dessas regras;
- (ii) regras baseadas na rede: somente considera informações contidas na representação do projeto como rede. A Tabela 4.7 exemplifica tais regras.
- (iii) regras baseadas no caminho crítico: são aquelas regras baseadas no cálculo do caminho crítico. Algumas dessas regras são exemplificadas na Tabela 4.8;
- (iv) regras baseadas em recursos: consideram somente informações provenientes da utilização de recursos. A Tabela 4.9 apresenta algumas dessas regras;
- (v) regras compostas: combinam informações das classificações anteriores. A Tabela 4.10 exemplifica algumas dessas regras.

Tabela 4.6: Exemplos de Regras de prioridade baseadas em atividades

Regra de Prioridade	Objetivo	$v(j)$
<i>Random</i> - RAND	-	-
<i>Minimum(Maximum) average job duration</i> - MAJD	min(max)	$\sum_{m \in M_j} \frac{d_{jm}}{ M_j }$
<i>Shortest(longest) processing time</i> - SPT(LPT)	min(max)	$d_{jm}, m \in M_j$

Tabela 4.7: Exemplos de Regras de prioridade baseadas na representação de rede

Regra de Prioridade	Objetivo	$v(j)$
<i>Maximum(Minimum) total of immediate successors</i> - MTS	max(min)	$ S_j $
<i>Minimum topological order</i> - MTO	min	j

Tabela 4.8: Exemplos de Regras de prioridade baseadas no caminho crítico

Regra de Prioridade	Objetivo	$v(j)$
<i>Minimum total slack</i> - SLK	min	$LFT_j - EFT_j$
<i>Minimum latest finish(start) time</i> - LFT(LST)	min	$LFT_j(LFT_j - d_j)$
<i>Minimum earliest finish(start) time</i> - EFT(EST)	min	$EFT_j(EFT_j - d_j)$

Tabela 4.9: Exemplos de Regras de prioridade baseadas em recursos

Regra de Prioridade	Objetivo	$v(j)$
<i>Total resource relative demand</i> - TRD	min	$\sum_{k=1}^K r_{jk}$
<i>Dynamic relative demand</i> - DRD	min	$\sum_{k=1}^K r_{jk} / \max\{r_{j'k} : j' \in D_n\}$
<i>Total resource scarcity</i> - TRS	min	$\sum_{k=1}^{K^p} r_{jk} / R_k^p$

Tabela 4.10: Exemplos de Regras de prioridade compostas

Regra de Prioridade	Objetivo	$v(j)$
<i>Maximum average resource demand</i> - MARD	max	$\sum_{m \in M_j} \frac{d_{jm}}{ M_j } \times \sum_{m \in M_j} \sum_{k=1}^{K^p} \frac{r_{jmk}^p}{(M_j R_k^p)}$
<i>Weighted resource utilisation ratio and precedence</i> - WRUP	max	$w_p S_j + (1 - w_p) \sum_{k=1}^{K^p} \frac{r_{jk}}{R_k^p}$
<i>Maximum remaining work</i> - RWK	max	$\min\{d_j : j \in D_n\} + S_j $
<i>Total resource relative demand</i> - TRD	min	$\sum_{k=1}^{K^p} r_{jk}$

Nessa dissertação não será diferenciada essa classificação. Como o objetivo é atribuir uma ordem para as atividades, essas regras serão discutidas, apenas, para essa finalidade. Logo, serão aqui tratadas como regras de prioridade relacionadas a atividades.

Talbot (1982) apresentou a primeira heurística baseada em regras de prioridade para o MRCPSP. Nesse trabalho foram utilizadas oito regras, como MAJD, LPT, LFT, EFT, MARD e RAND, como o primeiro estágio de um algoritmo B&B. O autor destaca a regra LFT como aquela que apresentou melhores resultados.

Em Boctor (1993), como já comentado no item de seção anterior, o autor apresentou um estudo estatístico comparando várias combinações de pares de regras de prioridade atividade-modo. Foram verificadas um total de sete regras para atividade, nas quais se encontravam as regras SLK, LFT, MTS, RWK, SPT e LPT. Boctor (1993) concluiu que as regras SLK e LFT obtiveram os melhores resultados.

Drexl e Gruenewald (1993) utilizaram uma variação estocástica da regra LFT, denominada no trabalho como STOCOM2.

No trabalho de Słowiński et al. (1994), foram utilizadas um total de doze regras de prioridade para atividades, dentre elas LFT, LST, EFT, EST, RWK, SPT e TRD, hibridizadas com a metaheurística SA, em uma variação multiobjetivo do MRCPP.

Özdamar (1999) hibridizou um total de nove regras de prioridade com um AG. O autor sugere que, ao utilizar apenas uma regra de prioridade, em algum instante essa regra irá apresentar desvantagem no término do sequenciamento. Então, justifica-se o uso de várias regras para exploração do domínio do problema. As regras utilizadas foram SLK, LFT, LST, SPT, WRUP, MTS, EST, EFT e RAND.

Dentre os trabalhos mais recentes e destacados pela literatura, Hartmann (2001), Alcaraz et al. (2003), Lova et al. (2009) agregam somente a regra LFT. Já Jozefowska et al. (2001) usaram somente a regra MTO. Bouleimen e Lecocq (2003), Jarboui et al. (2008), Damak et al. (2009), Van Peteghem e Vanhoucke (2010), Wang e Fang (2012) somente utilizam a regra RAND. Buddhakulsomsiri e Kim (2007) realizaram comparações entre pares de regras atividade-modo e concluíram que as regras SLK, LFT e LST apresentam melhores resultados. Van Peteghem e Vanhoucke (2009) utilizaram as regras LFT, LST, SLK e RWK. Já em Li e Zhang (2013) foi utilizada somente a regra SLK. Em todos esses trabalhos, as regras de prioridade são utilizadas na construção de uma solução inicial.

Métodos que utilizam um dos SGSs e uma regra de prioridade de forma determinística são conhecidos como abordagem *single-pass* e geram apenas um sequenciamento. Essa abordagem é a simples aplicação do SSS ou PSS com uma regra de prioridade. Por outro lado, procedimentos que visam gerar mais de um sequenciamento devem realizá-lo por alteração na regra de prioridade e várias execuções de algum SGS, sendo esses procedimentos conhecidos como abordagem *multi-pass* ou *X-pass*. Logo, a abordagem *multi-pass* utiliza um dos SGSs e várias regras de prioridade para gerar vários sequenciamentos. Os dois tipos de procedimentos *multi-pass* apresentados por Kolisch (1996b) são caracterizados como *multi-priority rule*, que utiliza um SGS e várias regras de prioridade de forma determinística, e *sampling*, que utiliza um SGS e uma (ou mais) regra de prioridade, porém os diferentes sequenciamentos são obtidos por um dispositivo aleatório aplicado sobre a regra de prioridade.

Kolisch (1996b) apresenta uma comparação entre os métodos SSS e PSS utilizando algumas das regras de prioridade listadas nas tabelas anteriores. Outros trabalhos em Kolisch e Drexl (1996), Kolisch (1996a), Lova et al. (2006) e Browning e Yassine (2010) também apresentam resultados na utilização de um SGS e regras de prioridade. Em todos esses trabalhos, as regras de prioridades que se destacam são extensões das regras LFT (LST) e SLK, ou seja, são as regras que apresentam melhores soluções ou sequenciamentos. Também, há destaque para o método *sampling*.

4.5.2 Métodos multi-pass

Procedimentos *multi-pass* são executados X vezes para gerar uma amostra com X soluções viáveis, sendo que a melhor solução é escolhida como o melhor sequenciamento (Kolisch, 1996b). Para gerar o sequenciamento, é utilizado um procedimento

aleatório com uma regra de prioridade sobre um SGS. Conforme exposto em [Kolisch \(1996b\)](#), a utilização de um procedimento aleatório pode ser interpretado como um mapeamento $\psi : j \in D_n \rightarrow [0, 1]$, dado que, a cada estágio n , é atribuída a cada atividade em D_n uma probabilidade $\psi(j)$ de seleção, com $\sum_{j \in D_n} \psi(j) = 1$. Três métodos diferentes para *Multi-pass sampling* são apresentados em [Kolisch e Drexel \(1996\)](#): (i) *Random sampling*; (ii) *Biased random sampling*; e (iii) *Regret based biased random sampling*. Os três métodos são descritos formalmente a seguir e um comentário a seu respeito finaliza esta subseção.

Random sampling : Atribui a cada atividade $j \in D_n$ a mesma probabilidade de seleção:

$$\psi(j) = \frac{1}{|D_n|} \quad (4.0)$$

Biased random sampling (BRS) : Cria uma probabilidade tendenciosa, dependente do valor de prioridade, para favorecer as atividades que aparentam ser a melhor escolha. O mapeamento é tal que:

$$\psi(j) = \frac{\frac{1}{v(j)}}{\sum_{i \in D_n} \frac{1}{v(i)}} \quad (4.0)$$

[Kolisch e Drexel \(1996\)](#) apontam que, para o método *Random sampling*, o valor de prioridade de uma atividade é ignorado. Já para o método *Biased random sampling*, existem duas desvantagens na utilização do mapeamento dado pela Equação (4.5.2). Primeiro, somente é possível utilizar as regras de prioridade que sempre atribuem um valor positivo para as atividades no conjunto de decisão. Segundo, se os valores de prioridade são altos, então o mapeamento se torna o mesmo dado pela Equação (4.5.2), independentemente do valor de prioridade de cada atividade.

Regret based biased random sampling (RBBRS) : Proposto por [Drexel \(1991\)](#), realiza uma comparação entre os valores de prioridade das atividades em relação ao melhor valor de prioridade e, em seguida, cria uma probabilidade baseada nessa comparação. Então, é atribuído, a cada atividade $j \in D_n$, um valor de prioridade $v(j)$, por meio de um objetivo O , indicando se a atividade do conjunto de decisão deve ser selecionada pelo mínimo $O = \min$, ou pelo máximo $O = \max$ valor de prioridade. Após, ϱ_j , conhecido como *regret*, compara o valor de prioridade da atividade j , de acordo com:

$$\varrho_j = \begin{cases} \max_{i \in D_n} \{v(i) - v(j)\}, & \text{se } O = \min \\ v(j) - \min_{i \in D_n} v(i), & \text{se } O = \max \end{cases} \quad (4.0)$$

Então, a probabilidade para escolher uma atividade $j \in D_n$ é definida como:

$$\psi(j) = \frac{(\varrho_j + 1)^\alpha}{\sum_{i \in D_n} (\varrho_i + 1)^\alpha} \quad (4.0)$$

Conforme observado por [Kolisch e Drexel \(1996\)](#), adicionando 1 a ϱ_j , é assegurado um valor diferente de zero para cada atividade e, então, qualquer sequenciamento pode ser gerado pelo SGS. Já [Kolisch e Drexel \(1996\)](#) discutem que o parâmetro α controla a probabilidade de escolha de uma atividade. Para altos valores de α , a Equação (4.5.2) vai se tornando determinística e se aproximando de uma abordagem *single-pass*, enquanto para o valor α igual a zero, será dada a característica de probabilidade uniforme para todas as atividades, de acordo com a Equação (4.5.2).

4.6 Heurísticas de Refinamento

Existem dois procedimentos heurísticos comuns na literatura utilizados como busca local para o MRCPSP. O primeiro, descrito inicialmente como *bounding rule* no algoritmo exato apresentado em [Sprecher et al. \(1997\)](#), foi adaptado como busca local em [Hartmann \(2001\)](#). Essa heurística é baseada no conceito de *Multi-mode Left Shift - MMLS*. Já o segundo, inicialmente proposto em [Li e Willis \(1992\)](#), para o RCPSP e, também utilizado com sucesso por [Özdamar e Ulusoy \(1996\)](#), foi adaptado para o MRCPSP, por [Lova et al. \(2009\)](#). Essa técnica utiliza construções sucessivas *forward* e *backward* para sequenciamentos. Esses procedimentos de busca local são descritos em seguida.

4.6.1 Multi-mode Left Shift Improvement - MLSI

Um MMLS em uma atividade j é uma operação em um sequenciamento que reduz o tempo de término de j sem alterar os modos ou tempos de término das outras atividades no sequenciamento. Conforme [Hartmann \(2001\)](#), duas características tornam esse movimento promissor: (i) MMLS considera tempos de início das atividades e modos, simultaneamente; e (ii) MMLS não deteriora a qualidade de uma solução, ou seja, a solução permanece viável e o *makespan* da solução não aumenta.

Há duas maneiras de construir a heurística MLSI: utilizando uma abordagem *single-pass* ou uma abordagem *multi-pass*. Ambas são descritas em [Hartmann \(2001\)](#), na presente dissertação apenas a abordagem *multi-pass* é considerada.

Seja j uma atividade do projeto já sequenciada, m o modo atual dessa atividade e πR_{kt} o restante disponível do recurso k no período t , definido como $\pi R_{kt} = R_k^\rho - \sum_{j \in A_t} r_{jmk}^\rho$, sendo A_t o conjunto das atividades consumindo recursos no período t . Então, a heurística de busca local MLSI é apresentada pelo Algoritmo 4.

O Algoritmo 4 descreve a abordagem *multi-pass*, pois todos os modos de uma atividade são verificados, ao contrário da abordagem *single-pass* que finaliza a verificação, para uma atividade, assim que o primeiro modo satisfaça a condição na linha 7.

O Algoritmo 4 busca melhorar o *makespan* de uma solução da seguinte maneira. Para cada atividade $j = 2, \dots, |J|$ do sequenciamento, libera-se o consumo de recur-

Algoritmo 4: Busca Local MLSI

```

1 para  $j \leftarrow 2$  até  $|J|$  faça
2    $\pi R_{kt} \leftarrow \pi R_{kt} + r_{jmk}^\rho, t = FT_{jm} - d_{jm}, \dots, FT_{jm}, k = 1, \dots, K^\rho$  ;
3   para  $\bar{m} \leftarrow 1$  até  $|M_j|$  faça
4     se  $\bar{m} \neq m$  && modo viável em relação a recursos não renováveis
       então
5        $EFT_j \leftarrow \max\{FT_i : i \in \mathcal{P}_j\} + d_{j\bar{m}}$  ;
6        $FT'_{j\bar{m}} \leftarrow \min\{t : EFT_{j\bar{m}} \leq t \leq T, r_{j\bar{m}k}^\rho \leq \pi R_{k\tau}, \tau =$ 
          $t - d_{j\bar{m}}, \dots, t, k = 1, \dots, K^\rho\}$  ;
7       se  $FT'_{j\bar{m}} \leq FT_{jm}$  então
8          $FT_{jm} \leftarrow FT'_{j\bar{m}}$  ;
9          $m \leftarrow \bar{m}$  ;
10      fim
11    fim
12  fim
13   $\pi R_{kt} \leftarrow \pi R_{kt} - r_{jmk}^\rho, t = FT_{jm} - d_{jm}, \dots, FT_{jm}, k = 1, \dots, K^\rho$  ;
14 fim

```

sos de j pelo modo corrente m , de acordo com a linha 2 e é verificado se um MMLS pode ser realizado. Assim, os modos M_j são testados. Para cada atividade, o modo $\bar{m}$, $\bar{m} \neq m$, viável em relação aos recursos não renováveis, é atribuído à atividade, caso satisfaça a condição da linha 7. Após o término dessa heurística, o resultado é um sequenciamento viável, com *makespan* menor ou igual ao *makespan* anterior. De acordo com Hartmann (2001), os modos de execução de todas as atividades devem estar ordenados de forma não decrescente, em relação ao tempo de duração, para que o modo com menor duração seja verificado primeiro. Experimentos preliminares realizados por Hartmann (2001) mostraram que o ganho na qualidade das soluções obtidas pelo método *multi-pass* em relação ao *single-pass* não vale o esforço computacional do primeiro. Um exemplo da execução da heurística MLSI como *single-pass* é dado pela Figura 4.4.

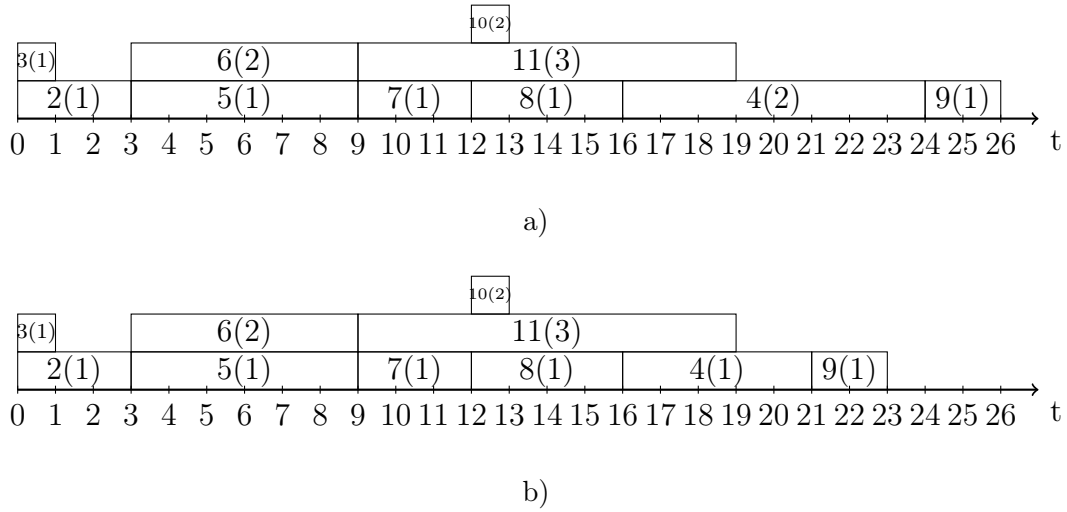


Figura 4.4: Exemplo de aplicação da busca local *single-pass* MLSI: (a) sequenciamento inicial. (b) sequenciamento final

Utilizando o sequenciamento apresentado pela Figura (4.4 a) como solução corrente, ao iniciar a heurística *single-pass*, as atividades 3 e 2 não realizam o movimento MMLS, pois já possuem os modos com menor duração. A atividade 5 possui apenas um modo e a troca de modo da atividade 6 torna a solução inviável em relação aos recursos não renováveis. Ao verificar a possibilidade de executar o MMLS nas atividades 7, 10, 8 e 11, constata-se que não é possível, devido já possuírem o modo com menor duração ou se tornarem inviáveis em relação aos recursos não renováveis. Então, verifica-se a atividade 4. Para essa atividade, é possível realizar o MMLS para o modo 1, reduzindo seu tempo de término para 21. Consequentemente, a atividade 9, que já possui o modo com menor duração, poderá começar no tempo 21 e finalizar no tempo 23. Assim, conclui-se a execução da heurística MLSI, obtendo o sequenciamento dado pela Figura (4.4 b). Percebe-se que o *makespan* anterior de valor 26 foi reduzido para 23 unidades de tempo.

4.6.2 Multi-mode forward/backward improvement - MFBI

Os sequenciamentos discutidos nos Algoritmos 2 (SSS) e 3 (PSS) possuem a característica de atribuírem tempos de início mais cedo possível para as atividades. Essa construção é conhecida pela literatura como sequenciamento *forward*. Percebe-se, então, que abordagens heurísticas para o RCPSP são executadas em dois estágios:

- i) uma lista de atividades é construída por meio de algum critério heurístico;
- ii) cada atividade, da lista de atividades, recebe tempo de início por meio da ordem apresentada na lista.

Pelo segundo estágio, de acordo com Li e Willis (1992), existem duas técnicas que podem ser utilizadas, *Forward-loading* e *Backward-loading*, sendo que, para a primeira, as atividades começam o mais cedo possível, já, para a segunda técnica, as atividades começam o mais tarde possível. Li e Willis (1992) observaram que, em alguns casos, a abordagem *Backward-loading* constrói sequenciamentos com menor

duração. Logo, Li e Willis (1992) definiram um método iterativo de sequenciamento, que utiliza as construções iterativas *forward* e *backward* como um procedimento *multi-pass*, que é executado até que não é mais possível melhorar a duração do projeto. A abordagem *single-pass* executa uma iteração *backward/forward* para tentar diminuir o *makespan* de um sequenciamento.

O processo iterativo é definido em Li e Willis (1992) pelos passos:

- i) Uma lista de atividades é definida e um sequenciamento *forward* é construído, de forma a atribuir tempo de início mais cedo possível para as atividades. A duração do projeto C_{max}^f obtida é, então, utilizada como o limite corrente;
- ii) As atividades são ordenadas de forma crescente pelos tempos de término, a partir do sequenciamento *forward* anterior. Então, percorrendo a lista de atividades da última atividade até a primeira, atribui-se tempo de término FT_j para a atividade j o mais tarde possível, de acordo com a Equação (4.6.2), e, consequentemente, atribui-se tempo de início $ST_j = FT_j - d_j$. No fim, obtém-se C_{max}^b , tal que $C_{max}^b = C_{max}^f - ST_1$, sendo 1 a primeira atividade do projeto. Se $C_{max}^b = C_{max}^f$, não é possível melhorar a duração do projeto e o processo iterativo é finalizado, caso contrário, $C_{max}^b < C_{max}^f$ e o passo (iii) é executado;
- iii) Novamente, as atividades são ordenadas de forma crescente, porém, pelos tempos de início, a partir do sequenciamento *backward* anterior. Então, percorrendo a lista de atividades da primeira atividade até a última, executa-se o passo 1, obtendo-se o novo C_{max}^f . Se $C_{max}^f = C_{max}^b$, não é possível melhorar a duração do projeto e o processo iterativo é finalizado; caso contrário, executa-se, novamente, o passo (ii).

$$FT_j = \begin{cases} C_{max}, & \text{se } S_j = \{\} \\ \min\{ST_i : \forall i \in S_j\}, & \text{caso contrário} \end{cases} \quad (4.0)$$

sendo S_j o conjunto das atividades ainda não sequenciadas.

No fim da execução do método iterativo *forward/backward*, o sequenciamento final é definido pela última construção *forward* e a duração final do projeto C_{max} será o último C_{max}^f obtido.

Esse método é aplicado ao RCPSP pelo algoritmo SSS. Posteriormente, Özdamar e Ulusoy (1996) adaptam o método *forward/backward* para o algoritmo PSS. Valls et al. (2005) criam o termo *justification* para descrever o método iterativo *forward/backward* e apresenta a capacidade desse método de melhorar a duração do projeto sem aumento significativo no tempo computacional e concluem que essa heurística deve ser usada na resolução do RCPSP. Recentemente, Sonmez e Uysal (2014) aplicaram a heurística *forward/backward* para o RCPSP com múltiplos projetos.

Os trabalhos em Boctor (1996), Özdamar (1999), Alcaraz et al. (2003), Van Peteghem e Vanhoucke (2008), Tseng e Chen (2009) e Coelho e Vanhoucke (2011) utilizam construções *forward* e *backward* para a resolução do MRCPSp, criando, separadamente, sequenciamentos *forward* e *backward*. Porém, nenhum desses trabalhos consideraram os modos de execução durante o processo iterativo. Então, Lova et al. (2009) adaptaram a heurística *forward/backward* para o MRCPSp, definindo

a nova heurística *multi-mode forward/backward improvement method* (MFBI). Essa adaptação, nos passos descritos anteriormente, considera os modos de execução de uma atividade da seguinte maneira:

- Se o sequenciamento em construção é *forward*, então, para o passo (iii) descrito anteriormente, cada atividade j da lista recebe tempo de término para o modo viável de acordo com:

$$\begin{aligned} m_j &= \arg \min_{m \in M_j} FT_{jm} \\ &= \arg \min_{m \in M_j} \min \left\{ t : EFT_{jm} \leq t \leq C_{max}^b, r_{jmk}^\rho \leq \pi R_{k\tau}, \right. \\ &\quad \left. \tau = t - d_{jm}, \dots, t, k = 1, \dots, K^\rho \right\} \end{aligned} \quad (4-1)$$

- Por outro lado, se o sequenciamento em construção é *backward*, então, para o passo (ii) descrito anteriormente, cada atividade j da lista recebe tempo de término para o modo viável de acordo com:

$$\begin{aligned} m_j &= \arg \max_{m \in M_j} ST_{jm} \\ &= \arg \max_{m \in M_j} \max \left\{ t : 0 \leq t \leq LFT_{jm}, r_{jmk}^\rho \leq \pi R_{k\tau}, \right. \\ &\quad \left. \tau = t - d_{jm}, \dots, t, k = 1, \dots, K^\rho \right\} \end{aligned} \quad (4-2)$$

Os trabalhos em [Van Peteghem e Vanhoucke \(2010\)](#), [Wauters et al. \(2011\)](#), [Wang e Fang \(2011\)](#), [Wang e Fang \(2012\)](#), [Soliman e Elgendi \(2014b\)](#), [Soliman e Elgendi \(2014a\)](#) e [Okada et al. \(2014\)](#) implementam a heurística MFBI, porém, diferentemente de [Lova et al. \(2009\)](#), para cada atividade, existe uma probabilidade associada de alterar o modo de execução durante as passagens *forward* e *backward*, ou seja, existe uma probabilidade de executar as Equações (4-1) ou (4-2).

Para exemplificar a aplicação da heurística MFBI, a Figura 4.5 apresenta um sequenciamento obtido por essa heurística, por meio da solução apresentada pela Figura 4.2.

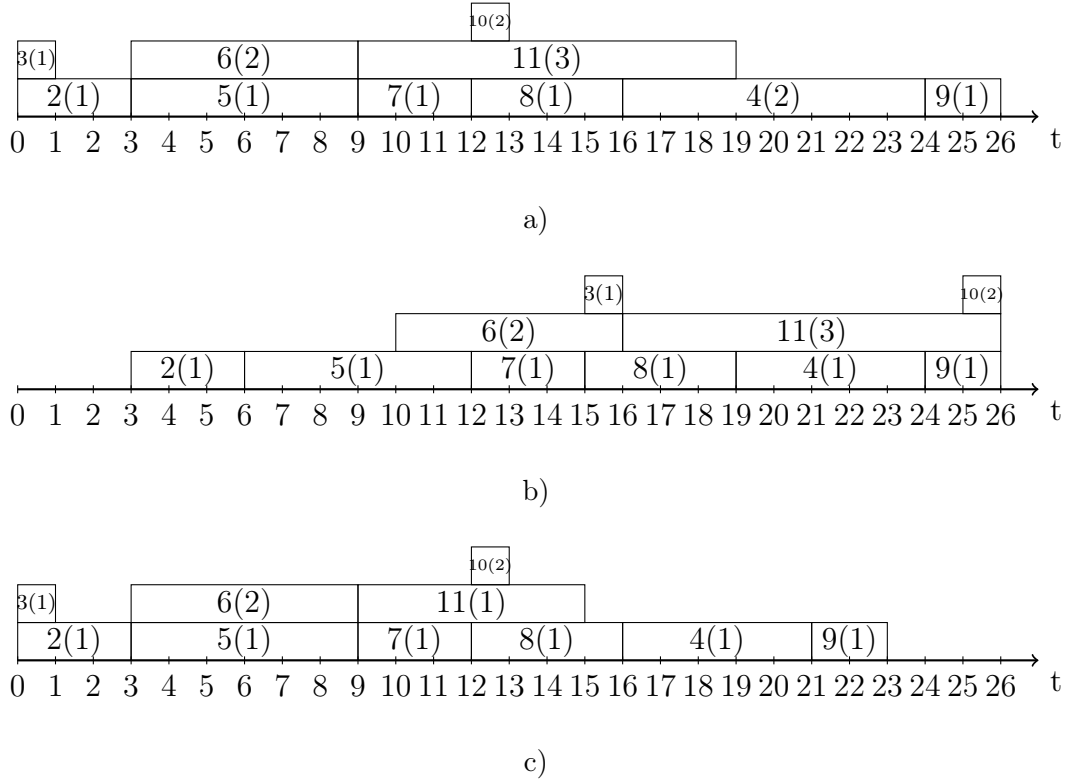


Figura 4.5: Exemplo de sequenciamento obtido pela heurística *multi-pass* MFBI: a) sequenciamento *forward* inicial; b) sequenciamento *backward*; c) sequenciamento *forward* final.

O sequenciamento *forward* inicial é dado na Figura (4.5 a), cujo *makespan* inicial é de 26 unidades e o consumo de recursos não renováveis é tal que $\sum_{j=1}^{12} r_{jm3}^{\nu} = 25$ e $\sum_{j=1}^{12} r_{jm4}^{\nu} = 39$. A ordem das atividades pelos tempos de término é 1, 3, 2, 5, 6, 7, 10, 8, 11, 4, 9, 12. Inicialmente, considera-se a atividade 9. O modo viável com menor duração é o modo corrente, portanto, esse modo é mantido. A próxima atividade a ser considerada é a atividade 4. Dentre os modos viáveis em relação aos recursos não renováveis, o modo 1 é determinado. Para a atividade 11, os modos 1 e 2 são inviáveis em relação aos recursos não renováveis, e então, o modo 3 permanece. Para as atividades em seguida, 8, 10, 7, 6, 5, 2 e 3, seus respectivos modos, também não são alterados. Ao fim da iteração *backward*, o sequenciamento na Figura (4.5 b) é obtido. O *makespan* é de 23 unidades, sendo 3 unidades menor do que o *makespan* inicial. Também houve alterações nos consumos de recursos não renováveis, sendo $\sum_{j=1}^{12} r_{jm3}^{\nu} = 27$ e $\sum_{j=1}^{12} r_{jm4}^{\nu} = 32$. As atividades são ordenadas, de forma crescente, pelo tempo de início. A ordem é 1, 2, 5, 6, 7, 3, 8, 11, 4, 9, 10, 12. A atividade 2 permanece com seu modo de execução, assim como as atividades seguintes 5, 6, 7, 3 e 8. Para a atividade 11, o modo é alterado para o novo modo 1. Após, não há alterações de modos das atividades seguintes. No fim, obtem-se o *makespan* de 23 unidades de tempo com consumo de recursos não renováveis igual a $\sum_{j=1}^{12} r_{jm3}^{\nu} = 27$ e $\sum_{j=1}^{12} r_{jm4}^{\nu} = 35$. Como não houve melhora no *makespan*, o processo é encerrado e o sequenciamento na Figura (4.5 c) passa a ser o novo sequenciamento, bem como a nova lista de atividades e modos são, respectivamente, $L_J = \langle 1, 2, 5, 6, 7, 3, 8, 11, 4, 9, 10, 12 \rangle$ e $M_J = \langle 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 1 \rangle$.

4.7 Iterated Local Search - ILS

Algoritmo 5: Algoritmo ILS

Saída: Solução: s^*

```

1  $s^* \leftarrow \text{GerarSolucaoInicial}$  ;
2  $s^* \leftarrow \text{BuscaLocal}(s^*)$  ;
3 repita
4    $s \leftarrow \text{Perturbacao}(s^*)$  ;
5    $s^{**} \leftarrow \text{BuscaLocal}(s)$  ;
6    $s^* \leftarrow \text{CritérioAceitacao}(s^*, s^{**})$  ;
7 até Critério de parada não satisfeito ;
```

Dado um conjunto de soluções X , é desejado explorar X movendo-se de uma solução $s \in X$ para outra que é *próxima* e *melhor* que s . De acordo com a descrição em [Lourenço et al. \(2003\)](#), a metaheurística ILS executa esse caminho de acordo com o Algoritmo 5.

Seja s^* a solução corrente. Aplica-se uma perturbação em s^* que a leva a outra solução $s \in X$. Após, uma busca local é realizada sobre s , conduzindo a $s^{**} \in X$. Se s^{**} passa por algum teste de aceitação, então ela será a nova solução corrente; caso contrário, o processo retorna para s^* e uma nova perturbação é executada. Segundo [Lourenço et al. \(2003\)](#), a metaheurística ILS deve produzir boas amostragens de ótimos locais se as perturbações não forem pequenas demais nem grande demais. Se a perturbação for pequena, o método sempre retornará para s^* e poucas soluções distintas serão exploradas; por outro lado, se a perturbação for grande demais, s será escolhida de forma aleatória e, portando, s^{**} poderá não representar boas amostragens de ótimos locais.

[Lourenço et al. \(2003\)](#) discutem a facilidade de adaptar a metaheurística ILS a vários problemas de otimização combinatória, assim como o sucesso da aplicação da metaheurística ILS a problemas de sequenciamento, como *Single Machine Total Weighted Tardiness Problem*, *Single and Parallel Machine Scheduling*, *Flow Shop Scheduling* e *Job Shop Scheduling*. Alguns estudos relacionados a *Project Scheduling* por meio da aplicação da metaheurística ILS são encontrados em [Ballestin e Trautmann \(2008\)](#) e [Hu et al. \(2013\)](#).

4.8 Clonal Selection Algorithm

O Algoritmo de Seleção Clonal (CLONALG), baseado em Sistemas Imunológicos Artificiais (SIA), proposto por [De Castro e Von Zuben \(2002\)](#), foi inicialmente apresentado para reconhecimento de padrões e, logo após, adaptado para resolução de funções multimodais, assim como problemas de otimização. O princípio da seleção clonal é usado para descrever as características de respostas em um sistema imunológico para antígenos. Esse sistema estabelece que somente células que respondem ou reconhecem os antígenos são selecionadas para proliferação, sendo que estas células fornecem as melhores respostas aos antígenos apresentados. As respostas que as células fornecem são conhecidas como anticorpos. Então, as células selecionadas passam por um processo de maturação, que aprimora sua afinidade em relação aos

antígenos apresentados.

Para a aplicação de um algoritmo baseado em Seleção Clonal, segundo [De Castro \(2006\)](#), alguns passos devem ser realizados:

1. Inicialização: Criação de uma população de anticorpos $POP(t)$;
2. Apresentação do antígeno: Avaliação da afinidade de cada anticorpo;
3. Seleção Clonal: Criação de uma população de clones $C(t)$ a partir da seleção dos melhores $N1$ anticorpos por meio da afinidade;
4. Maturação: Hipermutação dos clones em $C(t)$ proporcional a afinidade de cada clone;
5. Re-seleção: Os melhores anticorpos em $C(t)$ e $POP(t)$ são selecionados para uma nova população $POP(t + 1)$;
6. Diversidade: Introdução de $N2$ novos anticorpos em $POP(t + 1)$ substituindo anticorpos com menor afinidade.

De acordo com [De Castro \(2006\)](#), algumas observações são necessárias sobre o processo de seleção clonal, quando aplicadas a um problema de otimização:

- Para o passo 1, por simplificação, somente os anticorpos são apresentados para avaliação, sendo que cada anticorpo representa uma solução. No passo 2, não há antígeno para ser reconhecido, mas uma função objetivo a ser otimizada. Então, a afinidade do anticorpo corresponde à sua avaliação (*fitness*), segundo a função objetivo;
- Se o problema consiste em localizar múltiplos ótimos locais, então é preferível que os anticorpos selecionados para clonagem possuam o mesmo número de clones. A ideia é que cada anticorpo é visto como uma solução local, ou seja, se toda a população ou um subconjunto dela for selecionada para clonagem, cada anticorpo poderá representar uma solução em uma região do espaço de busca;
- A afinidade será utilizada para determinar a taxa de mutação de cada anticorpo, que é inversamente proporcional à sua afinidade, ou seja, quanto melhor o anticorpo responde à função, menor é a probabilidade (ou modificação) de hipermutação desse anticorpo.

[De Castro \(2006\)](#) apresenta o CLONALG, para problemas de otimização, de acordo com o Algoritmo 6. Nesse algoritmo, GER é um parâmetro que define o

número máximo de gerações geradas.

Algoritmo 6: Algoritmo CLONALG

Entrada: $GER, N1, N2, \beta$

Saída: POP

```

1 Inicializa( $POP$ ) ;
2 enquanto iteração atual  $t \leq GER$  faça
3    $f_{POP}(t) \leftarrow \text{AvaliarAnticorpos}(POP(t))$  ;
4    $C(t), f_C(t) \leftarrow \text{ClonarAnticorpos}(POP(t), N1, \beta, f_{POP}(t))$  ;
5    $C^*(t), f_{C^*}(t) \leftarrow \text{MutarAnticorpos}(C(t), f_C(t))$  ;
6    $POP(t+1) \leftarrow \text{SelecionarClones}(C^*(t), f_{C^*}(t))$  ;
7    $POP(t+1) \leftarrow \text{GerarAnticorpos}(N2)$  ;
8 fim
```

A aplicação de um algoritmo baseado em um SIA, de acordo com a literatura, mostra-se válida para a resolução de problemas de sequenciamento, tais como *Job shop* em [Hart et al. \(1998\)](#), [Coello et al. \(2003\)](#) e [Diana et al. \(2015\)](#), *Flow shop* em [Engin e Döyen \(2004\)](#) e [Sun e Yang \(2008\)](#), e *Project scheduling* em [Agarwal et al. \(2007\)](#), [Van Peteghem e Vanhoucke \(2009\)](#) e [Ju e Chen \(2012\)](#).

Capítulo 5

Algoritmos para a resolução do MRCPSP

Neste capítulo são descritas as implementações heurísticas utilizadas em conjunto com os algoritmos ILS e CLONALG. Inicialmente é detalhada a heurística construtiva baseada em regras de prioridade. Estruturas de vizinhança são apresentadas, devido a necessidade das metaheurísticas ILS e CLONALG modificarem uma solução por meio da perturbação ou mutação. Então, são apresentadas as propostas das metaheurísticas ILS e CLONALG implementadas nesta dissertação.

5.1 Construção de uma solução e esquemas de sequenciamento

De acordo com as regras de prioridade apresentadas na Seção 4.5 do Capítulo 4, assim como as abordagens *multi-pass*, nesta dissertação foram utilizadas as abordagens *Biased Random Sampling* (BRS), para determinar os modos de execução, e *Regret Based Biased Random Sampling* (RBBS), para determinar a ordem das atividades a ser considerada pelo sequenciador SSS. Foram utilizadas as regras SFM e MNDM para modos de execução; LFT e SLK para atividades, na construção de uma solução inicial. A regra MNDM, destacada na Tabela 5.1, é uma adaptação proposta nessa dissertação das regras Min-RDM e MNR, cujo valor de prioridade $v(m)$ atribui o modo com a menor relação de consumo de recursos por período. A motivação para a utilização da regra MNDM é devido essa regra explorar a duração das atividades, igualmente à regra SFM, entretanto verificando, também, o consumo de recursos.

Tabela 5.1: Regras de prioridade MNDM para modos de execução

Regra de Prioridade	Objetivo	$v(m)$
Minimum Normalized Demand Mode - MNDM	min	$d_{jm} \times \sum_{k=1}^{K^p} \frac{r_{jmk}}{R_k^p}$

Então, as listas L_J e L_M serão criadas pelo Algoritmo 7. Os passos presentes no Algoritmo 7 são descritos em seguida.

Algoritmo 7: Heurística contrutiva baseada em regras de prioridade

Saída: $s = \langle L_J, L_M \rangle$

- 1 Inicializar $n \leftarrow 1, L_J, L_M \leftarrow \emptyset$;
- 2 Inicializar $\pi R_{kt} \leftarrow R_k^\rho, t = 1, \dots, T, k = 1, \dots, K^\rho$;
- 3 Calcular $\{v(m) : \forall m, \forall j\}$, de acordo com as regras SFM ou MNDM ;
- 4 $M^* \leftarrow \{m_j^* : \forall j\}$, de acordo com BRS ;
- 5 **se** $ERR(M^*) > 0$ **então**
- 6 Reparar atribuição de modos ;
- 7 **fim**
- 8 Executar Algoritmo 1 ;
- 9 **enquanto** $|L_J| < |J|$ **faça**
- 10 Calcular D_n ;
- 11 Calcular $\{v(j) : j \in D_n\}$, de acordo com as regras LFT ou SLK ;
- 12 $L_J \leftarrow L_J \cup \{j^* : j \in D_n\}$, de acordo com RBBRS ;
- 13 $L_M \leftarrow L_M \cup \{m_j^*\}$;
- 14 $FT_{j^*} \leftarrow \min\{t : EFT_{j^*} \leq t \leq T, r_{j^*m_j^*k} \leq \pi R_{kt}, \tau = t - d_{j^*m_j^*}, \dots, t, k = 1, \dots, K^\rho\}$;
- 15 $\pi R_{kt} \leftarrow \pi R_{kt} - r_{j^*m_j^*k}, t = FT_{j^*} - d_{j^*m_j^*}, \dots, FT_{j^*}, k = 1, \dots, K^\rho$;
- 16 $n \leftarrow n + 1$;
- 17 **fim**

Inicialmente, são atribuídos modos de execução a todas as atividades, de acordo com as regras SFM ou MNDM, conforme a linha 3. As regras SFM ou MNDM são escolhidas aleatoriamente. Então, pela linha 4, é construído o conjunto M^* , que indica todos os modos de execução determinados para todas as atividades. Em seguida, o excesso de consumo de recursos não renováveis é verificado, de acordo com

$$ERR(M^*) = \sum_{j \in J} \max\{0, \sum_{k=1}^{K^\rho} r_{jmk} - R_k^\nu\} \quad (5.0)$$

Nessa Equação, se $ERR(M^*) = 0$, então a atribuição de modos é viável em relação aos recursos não renováveis. Isso significa que o total de recursos não renováveis consumido pelas atividades, na atribuição corrente de modos, não excede a capacidade máxima de cada recurso.

Se $ERR(M^*) > 0$, então a atribuição de modos é inviável em relação aos recursos não renováveis, e a atribuição corrente de modos passa por uma heurística de reparação, proposta por Hartmann (2001), descrita como: uma atividade real j do projeto é escolhida aleatoriamente, e seu modo m_j também é alterado para $\bar{m}_j$ de forma aleatória. Se $ERR(M^*)$ diminui, então $\bar{m}_j$ é atribuído à atividade j . O critério de parada adotado é dado por J tentativas seguidas sem sucesso de reparar a solução ou até que a solução seja reparada.

Definição das listas L_J e L_M pelo método RBBRS. Nessa fase é definida uma ordem para a lista L_J , a partir da atribuição de modos, e a lista de modos L_M é adequada de acordo com L_J . Ao término, a solução $s = \langle L_J, L_M \rangle$ é construída. Finalmente, todas as atividades são sequenciadas. O *loop* entre as linhas 9-16 do

Algoritmo 7 destaca o esquema de sequenciamento SSS.

Conforme já discutido, por meio do PSS não é garantido construir o sequenciamento ótimo. Essa foi a razão pela qual foi utilizado o SSS para sequenciar as atividades nesta dissertação.

Um exemplo de construção de uma solução, sobre o projeto apresentado na Figura 4.1 e Tabela 4.1, é apresentado em seguida.

Inicialmente calculam-se todos os valores de probabilidade $\psi(m)$ dos modos de execução, em relação às regras SFM e MNDM, para todas as atividades. Assume-se que a regra SFM foi escolhida para determinar todos os modos de execução, conforme destacados na Tabela 5.2.

Tabela 5.2: Cálculo dos valores de prioridade e probabilidade tendenciosa para a regra SFM e MNDM, por meio do método BRS. As células marcadas correspondem aos modos selecionados para cada atividade pela regra SFM.

j	m	$v(m)$		$\psi(m)$	
		SFM	MNDM	SFM	MNDM
2	1	3	2	0.750	0.714
2	2	9	5	0.250	0.286
3	1	1	1	0.455	0.402
3	2	1	0.778	0.455	0.517
3	3	5	5	0.091	0.080
4	1	5	3.889	0.615	0.578
4	2	8	5.333	0.385	0.422
5	1	6	1.333	1.0	1.0
6	1	2	0.444	0.75	0.75
6	2	6	1.333	0.25	0.25
7	1	3	1.667	0.727	0.727
7	2	8	4.444	0.273	0.273
8	1	4	2.667	0.556	0.417
8	2	10	3.333	0.222	0.333
8	3	10	4.444	0.222	0.25
9	1	2	0.444	0.673	0.507
9	2	7	0.778	0.192	0.29
9	3	10	1.111	0.135	0.203
10	1	1	0.44	0.474	0.5
10	2	1	0.5	0.474	0.444
10	3	9	4	0.053	0.056
11	1	6	3	0.441	0.339
11	2	9	2.5	0.294	0.407
11	3	10	4	0.265	0.254

Após a atribuição de modos para todas as atividades, é necessário construir a lista de atividades. A Tabela 5.3 apresenta o passo-a-passo da construção dessa lista.

Tabela 5.3: Cálculo dos valores de probabilidade para a regra LFT pelo método RB-BRS. As atividades j^* são aquelas escolhidas dentre todas as atividades disponíveis em D_n .

n	D_n	$LFT_j, j \in D_n$	$\psi(j), j \in D_n$	j^*	EFT_{j^*}	FT_{j^*}
1	{1}	0	1	1	0	0
2	{2,3,4}	3, 13, 17	0.714, 0.238, 0.048	2	3	3
3	{3,4,5,6}	13, 17, 9, 13	0.25, 0.05, 0.45, 0.25	3	1	4
4	{4,5,6}	17, 9, 13	0.067, 0.6, 0.333	5	9	9
5	{4,6,7,8}	17, 13, 17, 17	0.125, 0.625, 0.125, 0.125	6	5	6
6	{4,7,8,11}	17, 17, 17, 19	0.3, 0.3, 0.3, 0.1	11	11	12
7	{4,7,8}	17, 17, 17	0.333, 0.333, 0.333	8	13	13
8	{4,7}	17, 17	0.5, 0.5	7	17	21
9	{4,10}	17, 19	0.75, 0.25	4	5	26
10	{10,9}	19, 19	0.5, 0.5	9	19	28
11	{10}	19	1	10	18	27
12	{12}	19	1	12	19	28

Pela Tabela 5.3, no estágio $n = 1$, a atividade 1 é a primeira a entrar para a lista L_J . Então, as atividades 2, 3 e 4 estão disponíveis. No estágio $n = 2$, a atividade 2 é aquela que possui o menor tempo de término mais tardio dentre as atividades em D_n . Como já comentado, esse valor define o máximo tempo de término de uma atividade sem causar atraso no projeto. As probabilidades de escolha dessas atividades são, respectivamente, 71.4%, 23.8% e 0.48%. Foi, então, escolhida a atividade 2. Logo, estão disponíveis as atividades 3, 4, 5 e 6 para o próximo estágio. Determina-se a atividade 3. Ao término do processo, é construída a solução apresentada na Figura 5.1. Percebe-se que, em algum instante n da construção, a atividade j com menor probabilidade $\psi(j)$ pode ser escolhida, como é possível verificar no instante $n = 6$, a escolha da atividade 11 que possui a menor probabilidade de escolha. Também, verifica-se que as atividades 4, 7, 9 e 10 finalizaram nos tempos 26, 21, 28 e 27, respectivamente. Isso implicou em atraso, devido à indisponibilidade de recursos nas respectivas janelas $[0, 17]$, $[9, 17]$, $[17, 19]$ e $[17, 19]$.

L_J	1	2	3	5	6	11	8	7	4	9	10	12
L_M	1	1	2	1	1	1	1	2	1	1	1	1

Figura 5.1: Exemplo de solução inicial

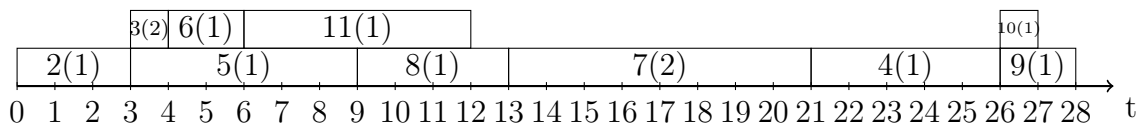


Figura 5.2: Sequenciamento obtido pelo Algoritmo 7

O sequenciamento referente a solução 5.1 é apresentado pela Figura 5.2.

A heurística MBFI é a busca local utilizada nesse trabalho, como tentativa de aprimorar o *makespan* da solução inicial. A literatura destaca essa heurística na resolução do MRCPSP, pois essa heurística pode diminuir o *makespan* de uma solução ao trocar as atividades de posição na lista L_J e também modificar os modos de execução. A cada iteração dessa heurística, duas execuções do SSS e duas ordenações pelo algoritmo Quicksort são necessárias, portanto, essa heurística foi executada como *single-pass*. Conforme as discussões no Capítulo anterior sobre a heurística MBFI quando as Equações (4-1) ou (4-2) são executadas, foi utilizada uma probabilidade associada para alterar o modo de execução durante as passagens *forward* e *backward*. Esse valor foi definido como 80% de probabilidade de verificar os modos de execução para cada atividade.

5.2 Estruturas de Vizinhança

Para Glover e Kochenberger (2003), vários problemas de otimização possuem características que os tornam difíceis em suas resoluções. Então, em várias situações, abordagens exatas não podem solucionar esses problemas em tempo aceitável (entenda-se polinomial) por algum algoritmo determinístico. Métodos conhecidos como busca local são bem recebidos pela comunidade científica devido a sua capacidade de encontrar boas soluções (ótimos locais) em curto tempo de busca (Papadimitriou e Steiglitz, 1998).

As técnicas de busca local são baseadas em uma ideia simples. Conforme descrito em Papadimitriou e Steiglitz (1998), dado um ponto viável $f \in F$, sendo F algum objetivo a ser otimizado de algum problema de otimização, é usual definir o conjunto $N(f)$ de pontos que estão próximos a f . Essa proximidade é definida como vizinhança. Dado que $f^1 \in N(f)$ tal que $c(f^1) \leq c(f), \forall f \in N(f)$, então f^1 é dito ser um ótimo local para a vizinhança $N(f)$. Para encontrar f^1 , é necessário definir movimentos que, a partir de um ponto f , são aplicados com o objetivo de explorar sua vizinhança, garantindo alcançar f^1 como ótimo local.

Existem dois tipos de vizinhanças para o MRCPSP:

- vizinhança baseada em atividades;
- vizinhança baseada em modos de execução.

Três movimentos para explorar o espaço de soluções nessas duas estruturas de vizinhança, são descritos a seguir.

Shift em uma atividade - N_1

A partir de uma lista de atividades, ao trocar uma atividade de posição, é necessário garantir que a relação de precedência dessa atividade seja respeitada. Então, sua posição pode ser alterada para trás ou para frente. Para uma atividade j , o predecessor e sucessor mais próximos, dados, respectivamente, como $p_j \in \mathcal{P}_j$ e $s_j \in \mathcal{S}_j$, são encontrados determinando-se o quão longe j pode mover-se para frente ou para trás. Assim, a atividade j pode movimentar-se para trás, até que p_j seja alcançado, ou para frente, até que s_j seja alcançado. A Figura 5.3 apresenta um exemplo desse movimento.

s												
L_J	1	2	5	7	6	3	8	4	11	10	9	12
L_M	1	1	1	1	2	1	1	2	3	2	1	1

s'												
L_J	1	2	4	5	7	6	3	8	11	10	9	12
L_M	1	1	2	1	1	2	1	1	3	2	1	1

Figura 5.3: Movimento N_1 aplicado à solução corrente s levando a um vizinho s'

Nesse movimento, a solução corrente é modificada escolhendo-se $j = 4$, encontrando seu predecessor e sucessor mais próximos, dados, respectivamente, como $p_4 = 1$ e $s_4 = 9$, e inserindo $j = 4$ em uma nova posição sem violação da restrição de precedência.

Swap entre atividades - N_2

Assim como no movimento N_1 , o movimento N_2 determina uma atividade j pela posição i e encontra seus predecessor, $p_j \in \mathcal{P}_j$, e sucessor, $s_j \in \mathcal{S}_j$, mais próximos, respectivamente nas posições i_p e i_s . Após, seleciona-se uma outra atividade $\bar{j}$ em uma posição $\bar{i}$ da lista L_J , tal que $\bar{j} \neq j$ e $i_p < \bar{i} < i_s$. Finalmente, troca-se as atividades j e $\bar{j}$ de posição se, e somente se, não há violação da restrição de precedência. A Figura 5.4 apresenta um exemplo desse movimento.

s												
L_J	1	2	5	7	6	3	8	4	11	10	9	12
L_M	1	1	1	1	2	1	1	2	3	2	1	1

s'												
L_J	1	2	5	7	6	4	8	3	11	10	9	12
L_M	1	1	1	1	2	2	1	1	3	2	1	1

Figura 5.4: Movimento N_2 aplicado à solução corrente s levando a um vizinho s'

A partir desse exemplo, ao selecionar a atividade $j = 4$ na posição $i = 8$, o predecessor e sucessor mais próximos, $p_4 = 1$ e $s_4 = 9$, em suas respectivas posições $i_p = 1$ e $i_s = 11$, indicam a distância que $j = 4$ pode mover-se para a esquerda ou direita. Então, seleciona-se $\bar{j} = 3$ na posição $\bar{i} = 6$. Como trocar as atividades $j = 4$ e $\bar{j} = 3$ de posição não implica em violação da restrição de precedência, o movimento N_2 é aplicado com sucesso. Por outro lado, caso $\bar{j} = 5$ na posição $\bar{i} = 3$, ao tentar trocar as atividades $j = 4$ e $\bar{j} = 5$ de posição, a restrição de precedência é violada, pois $\bar{j} = 5$ ficará à frente de dois de seus sucessores, $s_5 = 7$ e $s_5 = 8$ e então, o movimento N_2 não pode ser executado para um *swap* entre $j = 4$ e $\bar{j} = 5$.

Troca de um modo de execução - N_3

Dada uma lista de modos de execução L_M , esse movimento troca o modo de uma atividade quando possível. A Figura 5.5 apresenta uma solução s e um de seus

vizinhos s' ao aplicar o movimento N_3 sobre s , ao escolher a atividade 4 e trocar o modo de execução de 2 para 1.

s													
L_J	1	2	5	7	6	3	8	4	11	10	9	12	
L_M	1	1	1	1	2	1	1	2	3	2	1	1	

s'													
L_J	1	2	5	7	6	3	8	4	11	10	9	12	
L_M	1	1	1	1	2	1	1	1	3	2	1	1	

Figura 5.5: Movimento N_3 aplicado à s levando a um vizinho s' .

Descrição da Busca Local

A busca local baseada nos movimentos N_1 , N_2 e N_3 explora a vizinhança de uma solução s da seguinte forma. Inicialmente, o movimento N_2 é executado, determinando se é possível manter a viabilidade da lista de atividades; caso contrário, executa-se N_1 . Ambos os movimentos são executados, aleatoriamente, sobre a lista de atividades L_j . Após, o movimento N_3 é executado. Ao escolher, aleatoriamente, uma atividade $j \in L_J$, se j não é atividade crítica, então um novo modo m'_j , tal que $m'_j \neq m_j$ é determinado, aleatoriamente, para j . Por outro lado, se j é atividade crítica, então um modo m'_j é determinado para j , de acordo com a regra MNDM, por meio do método BRS. O movimento N_3 tende a atribuir o modo com menor relação de duração e consumo de recursos por período para atividades críticas.

5.3 Função de Avaliação

A função objetivo é frequentemente utilizada como função de avaliação, pois oferece uma medida de qualidade para uma solução. Quando o espaço de soluções inviáveis é considerado, a função de avaliação utiliza a função objetivo com o acréscimo de uma penalidade. Hartmann (2001) definiu uma função de avaliação considerando o *makespan* para soluções viáveis. Para as inviáveis, foi utilizado o horizonte do projeto, somado à quantidade de recursos não renováveis consumida em excesso. Essa mesma função de avaliação foi usada em Jozefowska et al. (2001). Entretanto, Alcaraz et al. (2003) apontaram que, usando essa avaliação, soluções inviáveis com *makespan* distintos e consumindo a mesma quantia em excesso de recursos não renováveis irão receber a mesma penalização. Para contornar tal problema, propuseram uma nova função de avaliação para soluções inviáveis. A função de avaliação $f(s)$ proposta em Alcaraz et al. (2003) é utilizada nesta dissertação e é dada como:

$$f(s) = \begin{cases} C_{\max}(s), & \text{se } ERR(M^*) = 0 \\ \max\{C_{\max}^{viavel}(POP)\} + C_{\max}(s) - CP_{\min} + ERR(M^*), & \text{caso contrário.} \end{cases} \quad (5.0)$$

Na Equação (5.3), $C_{\max}$ é o *makespan* da solução avaliada, $CP_{\min}$ é o caminho crítico minimal obtido pela atribuição dos modos viáveis com menor duração e o

Algoritmo 8: Algoritmo ILS-MRCPSP

Saída: Solução: s_{best}

```

1  $s^1 \leftarrow Inicializa()$  // de acordo com Alg. 7 ;
2  $s^1 \leftarrow BuscaLocal()$  // de acordo com MBFI ;
3  $s_{best} \leftarrow s^1$  ;
4 enquanto critério de parada não satisfeito faça
5   para  $k \leftarrow 1$  até  $K$  faça
6      $s^2 \leftarrow Perturbacao(s^1, k)$  // de acordo com  $N_2(N_1)$  e  $N_3$  ;
7      $s^2 \leftarrow BuscaLocal()$  ;
8     se  $f(s^2) < f(s_{best})$  então
9        $s^1 \leftarrow s^2$  ;
10       $s_{best} \leftarrow s^2$  ;
11       $k \leftarrow 1$  ;
12   fim
13 fim
14  $s^1 \leftarrow Inicializa()$  ;
15  $s^1 \leftarrow BuscaLocal()$  ;
16 se  $f(s^1) < f(s_{best})$  então
17    $s_{best} \leftarrow s^1$  ;
18 fim
19 fim

```

valor $\max\{C_{max}^{viavel}(POP)\}$ é o máximo *makespan* viável na população corrente, e $ERR(M^*$ definido pela Equação 5.1. Essa função é utilizada para avaliar as soluções em uma população. Como o ILS não é um algoritmo populacional, então o valor $\max\{C_{max}^{viavel}(POP)\}$ será modificado para o horizonte do projeto T , no ILS.

5.4 Adaptação da metaheurística ILS para o MRCPSP

A adaptação da metaheurística ILS proposta para a solução do MRCPSP é apresentada pelo Algoritmo 8, e é descrita a seguir.

Nesse algoritmo, o procedimento *Inicializa()* constrói uma solução de acordo com a discussão apresentada na Seção 5.1. O procedimento *BuscaLocal()* adotado é a heurística MBFI discutido, na seção 4.6.2. Já o procedimento *Perturbacao()*, altera a solução corrente de acordo com a descrição no item de seção 5.2. O parâmetro $k = 1, \dots, K$ define o grau de perturbação em uma solução, sendo K o máximo grau de perturbação. O grau de perturbação (modificação) em uma solução é dado pela quantidade de execuções dos movimentos N_2 (ou N_1) e N_3 , respectivamente, sobre as listas L_J e L_M . Percebe-se que essa adaptação do ILS possui o conceito de *multi-start* quando $k = K$ e não há melhora na solução.

A função que descreve o grau de modificação em uma solução é dada por:

$$\Theta = 1 + (((100 - \alpha) \times (MAX_MOD \times |J|))/100) \quad (5.0)$$

sendo $\alpha = 100 * e^{-2*\lambda}$ e $\lambda = (k - k^{min})/(k^{max} - k^{min})$. O valor λ apresenta uma normalização dos valores de k e indica que, quanto menor o valor de k , menor

será o valor de λ , sendo K inteiro positivo. O valor Θ indica a quantidade de movimentos N_2 (ou N_1) e N_3 que será executada sobre a solução corrente. Já o valor MAX_MOD define a perturbação máxima em uma solução com $|J|$ atividades reais. Esse valor representa a maior porcentagem de modificação em uma solução. A Equação (5.4) foi adaptada de [Van Peteghem e Vanhoucke \(2009\)](#). A relação entre Θ , k e MAX_MOD , descrita pela Equação (5.4) é apresentada na Figura 5.6 para $|J| = 10$.

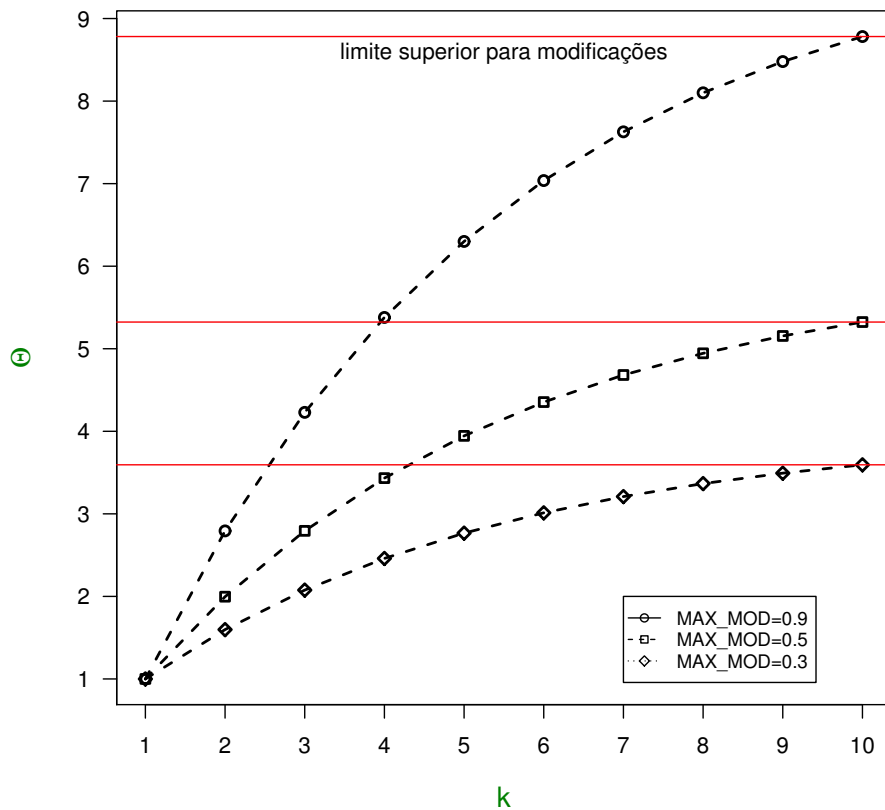


Figura 5.6: Número de modificações em função de k e MAX_MOD

De acordo com a Equação (5.4) e Figura 5.6, o valor MAX_MOD deve variar entre $[0.1, \dots, 1]$ e é representado pelas linhas horizontais, que descrevem a maior modificação em uma dada solução. Quanto menor o nível de perturbação k , menor será o número de modificações Θ em uma solução. Também é verificado que, quanto maior o valor MAX_MOD , maior é o número de modificações para valores iniciais de k .

Algoritmo 9: Algoritmo CLONALG-MRCPSP**Entrada:** $POP, N1, N2, \beta$ **Saída:** POP

```

1 Inicializa( $POP$ ) // de acordo com Alg. 7 ;
2  $POP \leftarrow BuscaLocal(POP)$  // de acordo com MBFI ;
3 enquanto critério de parada não satisfeito faça
4    $f_{POP}(t) \leftarrow AvaliarAnticorpos(POP(t))$  ;
5    $C(t), f_C(t) \leftarrow ClonarAnticorpos(POP(t), N1, \beta, f_{POP}(t))$  ;
6    $C^*(t), f_{C^*}(t) \leftarrow MutarAnticorpos(C(t), f_C(t))$  // por  $N_2(N_1)$  e  $N_3$  ;
7    $POP(t+1) \leftarrow SelecionarClones(C^*(t), f_{C^*}(t))$  ;
8    $POP(t+1) \leftarrow GerarAnticorpos(N2)$  // de acordo com Alg. 7 ;
9    $POP(t+1) \leftarrow BuscaLocal(POP(t+1))$  ;
10 fim

```

5.5 Adaptação da metaheurística CLONALG para o MRCPSP

A adaptação da metaheurística CLONALG proposta para a resolução do MRCPSP é apresentada pelo Algoritmo 9.

Os procedimentos *Inicializa()* e *BuscaLocal()* são os mesmos utilizados na adaptação da metaheurística ILS. O procedimento *GerarAnticorpos()* é idêntico ao procedimento *Inicializa()*. O procedimento *ClonarAnticorpos()* cria β cópias das $N1$ melhores soluções. Já o procedimento *MutarAnticorpos()*, presente na linha 6, é um procedimento de busca em vizinhança, que também é executado de acordo com a descrição no item de seção 5.2, porém, o grau de modificação de uma solução é inversamente proporcional à avaliação da solução, ou seja, quanto melhor uma solução, menor é a modificação realizada sobre essa solução.

A Equação (5.4) é utilizada para a mutação em uma solução, porém, alterando $\lambda = (f(s) - f(s)^{min}) / (f(s)^{max} - f(s)^{min})$, sendo $f(s)$ de acordo com a Equação (5.3). Logo, pelo valor λ , quanto menor a avaliação de uma solução $f(s)$, menor será o número de modificações em uma solução. Finalmente, o procedimento *SelecionarClones()* seleciona as melhores soluções na população de clones.

Esse algoritmo tende a manter *clusters* no espaço de busca e, ao longo das gerações, aqueles *clusters* com menores valores de avaliação serão substituídos por novas soluções.

Capítulo 6

Resultados Computacionais

Nesse capítulo são apresentados os testes computacionais realizados para os algoritmos propostos. Inicialmente é apresentado o planejamento estatístico para a análise de parâmetros dos algoritmos ILS e CLONALG. Após, são avaliados os resultados produzidos por esses algoritmos por meio de comparações entre os resultados. Ao final, os melhores resultados obtidos são apresentados para comparação entre os trabalhos mais recentes na literatura.

6.1 Descrição dos Experimentos

Todos os algoritmos utilizados nessa dissertação foram implementados em linguagem C usando compilador GNU GCC 4.8.2. Os testes foram realizados em um computador com processador Intel Core i3-3217U; 1.80GHz; 4GB RAM; e sistema operacional Ubuntu Linux 64 bits. Para os testes, foram utilizadas as classes de instâncias da biblioteca pública *Project Scheduling Problem Library* (PSPLIB) (cf. [Kolisch e Sprecher \(1997\)](#)). As características dessas classes são apresentadas na Tabela 6.1.

Tabela 6.1: Classes de instâncias PSPLIB

Classe	J10	J12	J14	J16	J18	J20	J30
# instâncias viáveis	536	547	551	550	552	554	552
# atividades	10	12	14	16	18	20	30
# modos de execução	3	3	3	3	3	3	3
# máximo de sucessores por atividade	3	3	3	3	3	3	3
# recursos renováveis	2	2	2	2	2	2	2
# recursos não renováveis	2	2	2	2	2	2	2

Para as classes J10-J20, valores ótimos são conhecidos. Para o conjunto J30, as

soluções foram encontradas apenas pela utilização de métodos heurísticos.

Para avaliar a resolução de cada grupo de instância foi utilizado o percentual do desvio médio (*GAP*) para as melhores soluções conhecidas, ou seja, soluções alcançadas por métodos exatos ou técnicas heurísticas. Esse valor é calculado como:

$$GAP = \frac{\bar{f} - f^*}{f^*} \times 100 \quad (6.0)$$

Para as classes J10-J20, o valor *GAP* indica o percentual do desvio da média das soluções $\bar{f}$ para 15 execuções em cada instância, em relação ao valor ótimo f^* conhecido na literatura. Já para a classe J30, *GAP* indica o percentual do desvio da média das soluções $\bar{f}$ para 15 execuções em cada instância, em relação ao limite inferior f^* conhecido.

O critério de parada adotado foi o número de sequenciamentos gerados, sendo esse igual a 5000. [Kolisch e Hartmann \(2006\)](#) definem que um sequenciamento corresponde, no máximo, a uma atribuição de tempo de início para cada atividade do projeto. Ou seja, para cada instante em que um modo viável é atribuído a uma atividade, essa operação corresponde à parcela $\frac{1}{|J|}$ de um sequenciamento, sendo $|J|$ o número de atividades do projeto.

A análise de parâmetros de ambos os algoritmo, ILS-MRCPSP e CLONALG-MRCPSP, será realizada pelos seguinte metodologia:

- (i) Cada instância da classe J20 foi solucionada 15 vezes, obtendo então o *GAP* de acordo com a Equação (6.1). Esse valor indica o *GAP* esperado para o algoritmo resolver a instância avaliada;
- (ii) Após resolver todas as instâncias, foi obtido o GAP_{avr} , que indica o *GAP* esperado para a resolução da classe J20. Esse valor é a variável de resposta, indicando uma observação para a análise estatística;
- (iii) Testes apontaram que uma única observação é necessária para alcançar normalidade nos dados, portanto, para realizar o teste ANOVA (cf. [Montgomery \(2013\)](#)) e identificar influência dos parâmetros, foram utilizadas duas observações, GAP_{avr}^1 e GAP_{avr}^2 . O teste ANOVA verifica a influência dos parâmetros principais, assim como interações entre os mesmos;
- (iv) Após o teste ANOVA, o teste de Tukey detalhado em [Montgomery \(2013\)](#) foi executado para informar qual(ais) nível(eis) de cada parâmetro, assim como cada combinação entre parâmetros, fornecem os melhores valores de *GAP*. Esse teste apresenta um intervalo de confiança para a diferença entre as médias de combinações de níveis entre parâmetros.
- (v) A análise de parâmetros, pela variável de resposta GAP_{avr} , indica quais os valores de níveis dos parâmetros se destacam na convergência dos algoritmos, bem como a tendência da variação dos parâmetros, sem a necessidade de testes exaustivos.

A validação do teste ANOVA depende de 3 premissas sobre os resíduos calculados: (i) normalidade; (ii) homoscedasticidade e (iii) independência ([Montgomery, 2013](#)). Para verificar essas premissas, os testes Shapiro-Wilk (cf. [Shapiro e Wilk](#)

(1965)), Bartlett (cf. [Montgomery \(2013\)](#)) e Durbin-Watson (cf. [J. Durbin \(1950\)](#)) foram realizados. As hipóteses nulas H_0 , para os três testes são dadas como:

- Shapiro-Wilk, H_0 : resíduos possuem distribuição normal;
- Bartlett, H_0 : resíduos possuem variâncias iguais;
- Durbin-Watson, H_0 : não há correlação entre os resíduos, i.e., são eventos independentes.

Todos os testes estatísticos nesta dissertação foram realizados utilizando o *software* R.

6.2 Análise de parâmetros do ILS-MRCPSP

O algoritmo ILS-MRCPSP possui 3 parâmetros: o número de iterações ($ITER$), o máximo nível de perturbação (K) e o percentual máximo de modificação em uma solução (MAX_MOD), sendo esse último parâmetro um limite superior para os valores de $k = 1, \dots, K$, correspondendo à máxima modificação MAX_MOD em uma solução. O valor para esse parâmetro indica o percentual aproximado de modificação em relação ao número de atividades no projeto. Por exemplo, $MAX_MOD = 0.3$ é uma aproximação de $30\%|J|$, para $|J|=20$, ou seja, corresponde a, aproximadamente, 6 modificações em uma solução com 20 atividades, quando a Equação (5.4) é utilizada para determinar a quantidade de modificações em uma solução.

Como o critério de parada adotado é o número de sequenciamentos gerados, o parâmetro $ITER$ não foi analisado, pois o algoritmo encerra a busca assim que 5000 sequenciamentos são gerados. Os níveis de cada parâmetro, bem como a ordem dos experimentos de cada combinação de nível, são apresentados pela Tabela 6.2. A ordem dos experimentos foi escolhida aleatoriamente.

Tabela 6.2: Ordem dos experimentos para cada combinação de níveis dos parâmetros

K	MAX_MOD			
	0.3	0.5	0.7	0.9
4	#14	#1	#5	#8
6	#2	#12	#6	#16
8	#13	#15	#10	#7
10	#11	#3	#9	#4

A Figura 6.1 apresenta a relação entre os parâmetros K e MAX_MOD , utilizando os níveis mínimo e máximo para MAX_MOD da Tabela 6.2.

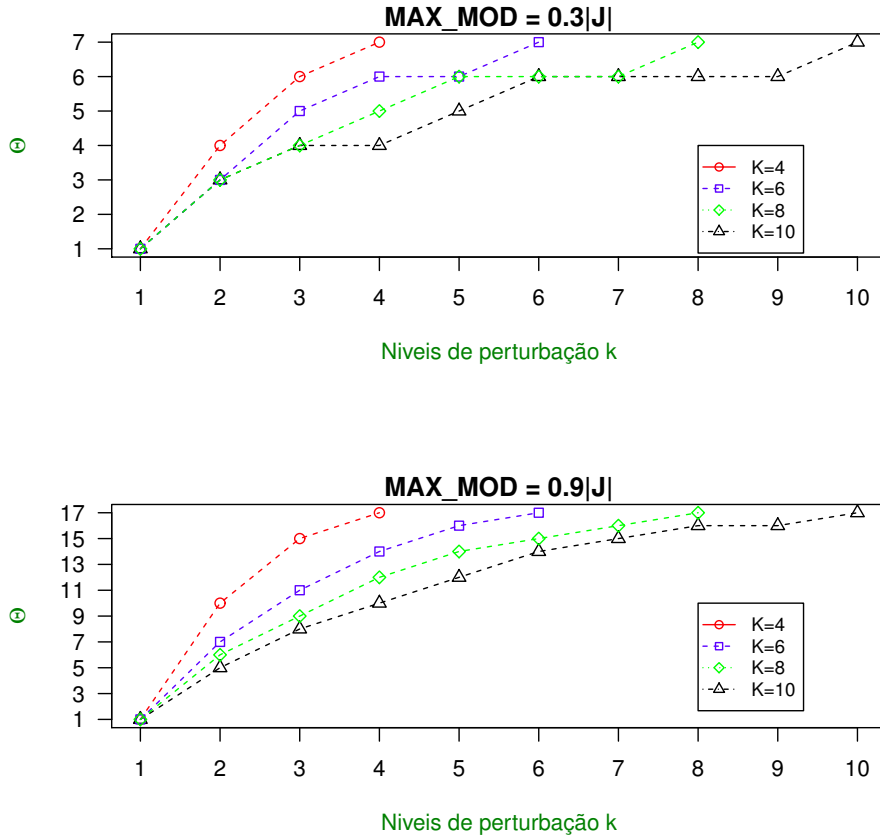


Figura 6.1: Número de modificações Θ em função dos níveis de perturbação $k = 1, \dots, K$ para máximas perturbações 0.3 e 0.9, de acordo com a Equação 5.4, sobre a classe J20.

De acordo com a Figura 6.1, o número de modificações cresce, rapidamente, para o valor $MAX_MOD = 0.9$ em relação a $MAX_MOD = 0.3$, ou seja, quanto maior o valor de MAX_MOD , maior é perturbação entre duas vizinhanças no início do processo de perturbação.

A Tabela 6.3 apresenta os valores obtidos para cada combinação de níveis de parâmetros. A tendência de cada parâmetro, de acordo com a variação de nível em relação ao GAP_{avr} , é mostrada na Figura 6.2.

Tabela 6.3: Valores GAP_{avr}^1 e GAP_{avr}^2 obtidos para cada combinação de parâmetro

Experimento	K	MAX_MOD	GAP_{avr}^1	GAP_{avr}^2
#1	4	0.5	1.7869	1.8163
#2	6	0.3	1.7399	1.7336
#3	10	0.5	1.7416	1.7256
#4	10	0.9	1.7749	1.7927
#5	4	0.7	1.8113	1.8352
#6	6	0.7	1.7999	1.8139
#7	8	0.9	1.7996	1.8055
#8	4	0.9	1.8092	1.8352
#9	10	0.7	1.7692	1.7456
#10	8	0.7	1.7703	1.7825
#11	10	0.3	1.6953	1.6971
#12	6	0.5	1.7896	1.7845
#13	8	0.3	1.7118	1.7108
#14	4	0.3	1.7765	1.7780
#15	8	0.5	1.7749	1.7491
#16	6	0.9	1.8309	1.7964

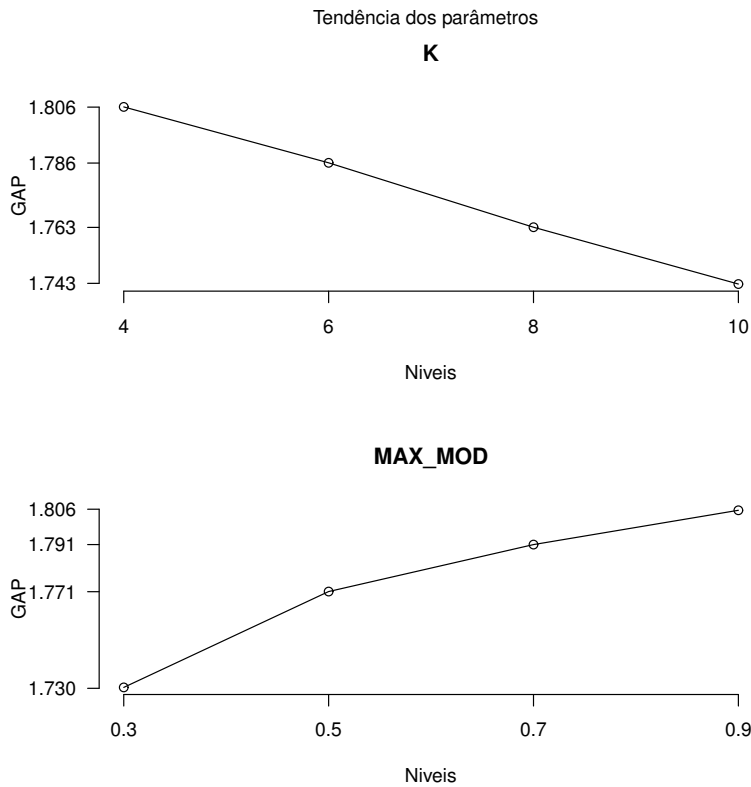


Figura 6.2: Tendência dos parâmetros do ILS-MRCPSP ao variar o nível.

De acordo com a Figura 6.2, o parâmetro K apresenta menor valor de GAP , em média, ao aumentar o nível. Isso significa que o ILS-MRCPSP oferece boas soluções quando a perturbação explora várias vizinhanças (cada vez maiores). Em relação

ao parâmetro MAX_MOD , o ILS-MRCPSP perde a capacidade de oferecer boas soluções com o aumento do nível desse parâmetro, sendo não desejável obter grandes modificações em uma solução.

Portanto, a análise da tendência desses parâmetros sugere criar vários níveis de perturbação para explorar o maior número possível de vizinhanças, até que a maior vizinhança indicada por MAX_MOD seja alcançada, para, então, reiniciar a solução corrente. A Tabela 6.4 apresenta a influência relativa dos parâmetros. Cada linha dessa tabela, de acordo com Roy (2010), indica a média dos valores da variável de resposta, obtida pelo nível de cada parâmetro. As melhores respostas estão destacadas em negrito.

Tabela 6.4: Respostas entre as médias dos níveis de parâmetros

Níveis		K	MAX_MOD
K	MAX_MOD		
4	0.3	1.806075	1.730375
6	0.5	1.786088	1.771062
8	0.7	1.763062	1.790987
10	0.9	1.742750	1.805550
Δ		0.063325	0.075175
Rank		2	1

Pela Tabela 6.4, o parâmetro MAX_MOD aparenta possuir maior influência do que o parâmetro K na convergência do ILS-MRCPSP, de acordo com o valor Δ , que refere-se à diferença entre as médias máxima e mínima dos valores de GAP_{avr} . Destaca-se o nível 10 do parâmetro k , assim como o nível 0.3 do parâmetro MAX_MOD . Esses níveis apresentam, em média, as melhores respostas. As Figuras 6.3 e 6.4 apresentam os boxplots para cada parâmetro em discussão. Já a Figura 6.5 apresenta a interação entre cada parâmetro ao variar os níveis.

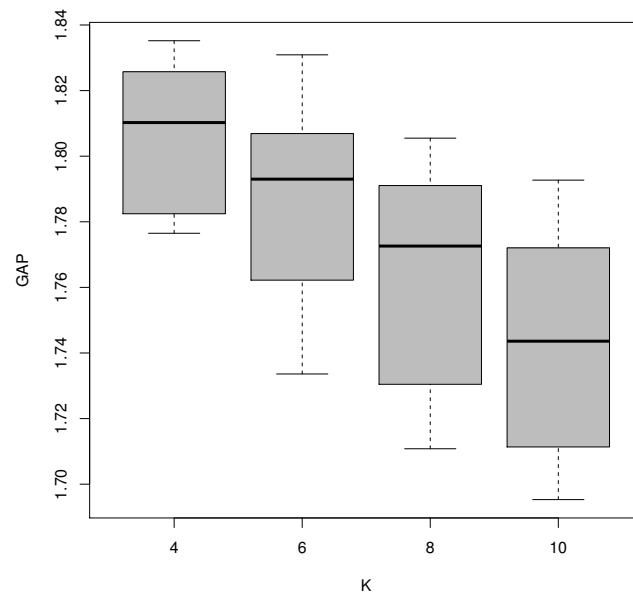


Figura 6.3: Boxplot para os níveis do parâmetro K em relação ao GAP

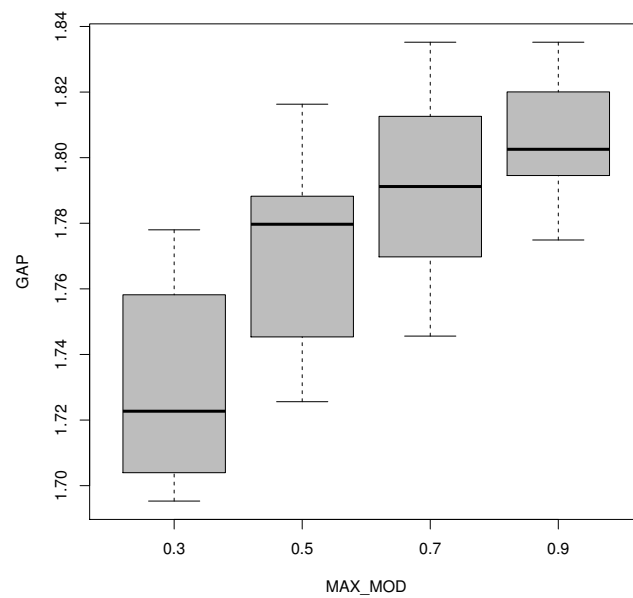


Figura 6.4: Boxplot para os níveis do parâmetro MAX_MOD em relação ao GAP

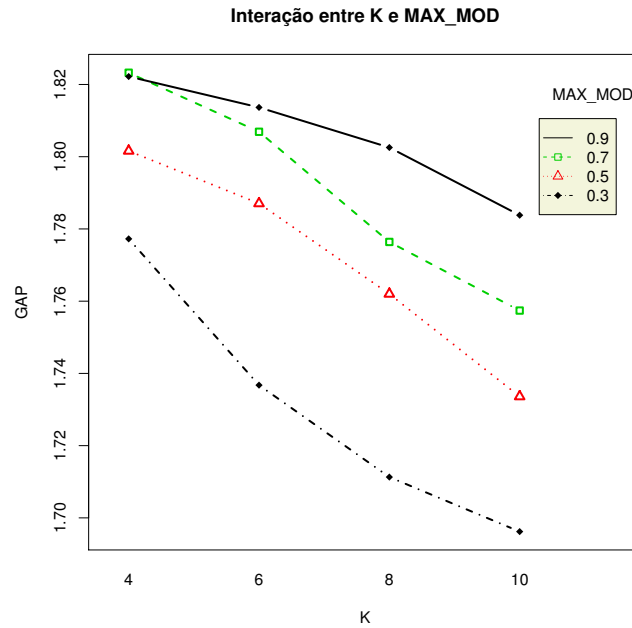


Figura 6.5: Interação entre os parâmetros K e MAX_MOD pela variação de cada nível

Ao analisar o boxplot da Figura 6.3, destacam-se os níveis 8 e 10 do parâmetro k , porque esses níveis apresentam as menores medianas. Entretanto, houve interseção entre as caixas. Logo, não é possível afirmar qual nível apresenta melhores respostas. Já para o boxplot da Figura 6.4 destacam-se os níveis 0.3 e 0.5 do parâmetro MAX_MOD . Entretanto, como já comentado, não é possível afirmar qual nível apresenta melhores respostas. O gráfico de interação entre os parâmetros k e MAX_MOD da Figura 6.5 indica interação mínima entre esses parâmetros, por meio dos níveis 0.9 e 0.7 do parâmetro MAX_MOD , quando o parâmetro K altera do nível 4 para nível 6.

Para confirmar a hipótese de diferença estatística entre os níveis de cada parâmetro e a interação entre os parâmetros, o teste ANOVA é realizado. O resultado é apresentado na Tabela 6.5. Esse teste indicou evidência estatística, com 95% de confiança, que permita afirmar que há diferenças entre as médias dos níveis dos parâmetros K e MAX_MOD , porém, não aponta interação entre os parâmetros.

Tabela 6.5: Tabela ANOVA para $\alpha = 0.05$

	DF	SS	MS	F_0	p_{valor}
K	3	0.018161	0.006054	34.910	2.98^{-7}
MAX_MOD	3	0.025558	0.008519	49.129	2.70^{-8}
$K : MAX_MOD$	9	0.001646	0.000183	1.054	0.443
Resíduos	16	0.002775	0.000173		

Os testes Shapiro-Wilk, Bartlett e Durbin-Watson apresentam valores p_{value} , respectivamente, 0.5028, 0.4687 e 0.922, para $\alpha = 0.05$. Esses valores não negam as hipóteses H_0 e, portanto, validam o teste ANOVA. Consequentemente, o teste

de Tukey é realizado para identificar quais níveis apresentam melhores respostas em cada parâmetro. As Figuras 6.6 e 6.7 apresentam os gráficos obtidos pelos testes de Tukey. Se o valor zero não pertence ao intervalo apresentado, então existe diferença entre os níveis. Quanto mais afastado de zero, maior é a diferença entre as médias dos níveis em comparação.

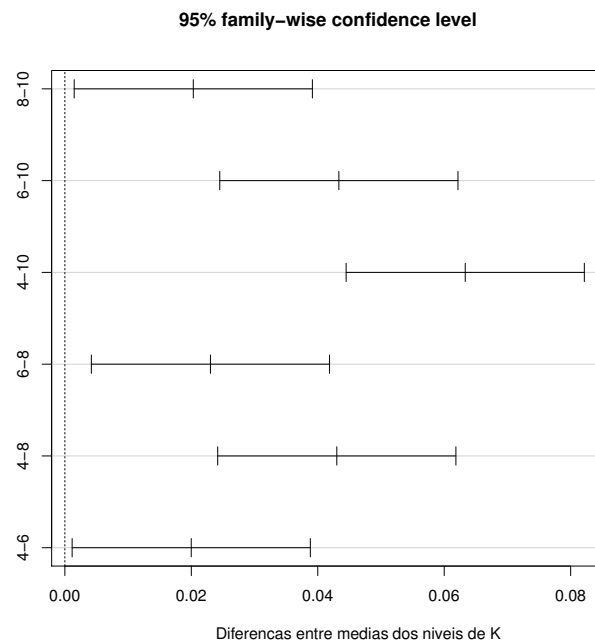


Figura 6.6: Diferenças entre as médias dos níveis do parâmetro K

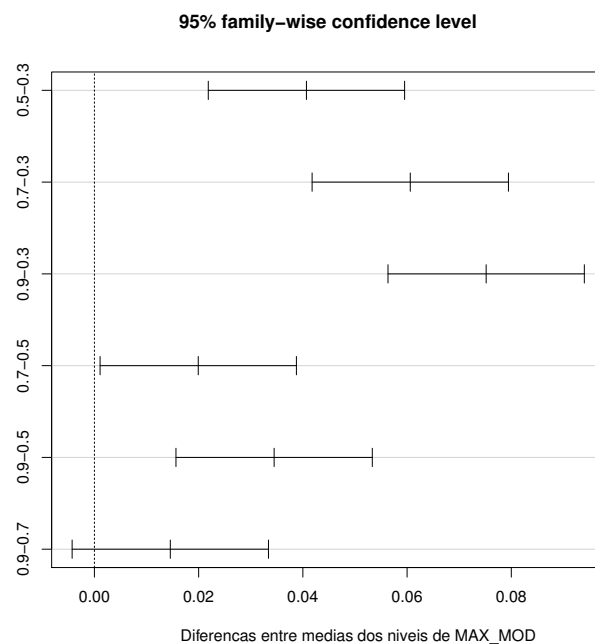


Figura 6.7: Diferenças entre as médias dos níveis do parâmetro MAX_MOD

Pela Figura 6.6, as maiores diferenças envolvem os níveis 4-10, 6-10, e 4-8 do parâmetro K . Como já comentado na análise do boxplot de K , o nível 10 foi aquele que apresentou a melhor mediana. Portanto, de fato, o nível 10 é aquele que apresentou os melhores valores de GAP , seguido do nível 8. Já o nível 4 foi aquele que apresentou valores altos de GAP , em média. Essa análise corrobora a discussão apresentada na tendência dos parâmetros, assim como no boxplot do parâmetro K .

Sobre a Figura 6.7, as maiores diferenças acontecem quando $MAX_MOD = 0.3$. Por ser o nível que apresentou a menor mediana pelo boxplot, é possível afirmar que esse nível apresentou os melhores valores de GAP , em conformidade com a análise de tendência dos parâmetros. Percebe-se que não há diferença estatística entre os níveis 0.7 e 0.9.

A análise discutida sobre os parâmetros K e MAX_MOD indica que níveis altos para o parâmetro K são desejáveis, ao contrário do parâmetro MAX_MOD , cujos níveis baixos são preferíveis para ajuste no ILS-MRCPSP.

6.3 Análise de parâmetros do CLONALG-MRCPSP

O algoritmo CLONALG-MRCPSP possui 5 parâmetros: (i) o número de gerações (GER), (ii) o tamanho da população (POP), (iii) a quantidade de soluções selecionadas para clonagem ($N1$), (iv) a quantidade de clones por solução (β) que passará pela mutação e (v) a quantidade de novas soluções inseridas para a próxima geração ($N2$).

No entanto, como o critério de parada adotado é o número de sequenciamentos gerados, a análise da influência do parâmetro GER não é realizada nessa seção. Assim, os valores de cada parâmetro analisado são apresentados pela Tabela 6.6.

Tabela 6.6: Ordem dos experimentos para cada combinação de valores dos parâmetros

		POP								
		100			200			300		
		N1								
β	N2	50	70	100	50	70	100	50	70	100
3	10	#24	#71	#59	#36	#57	#49	#56	#14	#37
	15	#22	#29	#33	#5	#35	#81	#13	#34	#39
	20	#63	#43	#58	#38	#78	#70	#9	#72	#80
5	10	#67	#76	#8	#30	#53	#79	#11	#6	#48
	15	#69	#19	#62	#74	#28	#25	#23	#73	#54
	20	#77	#61	#12	#7	#50	#45	#26	#40	#60
10	10	#75	#47	#17	#52	#21	#15	#20	#3	#10
	15	#16	#27	#44	#18	#51	#66	#2	#1	#55
	20	#64	#42	#31	#68	#32	#65	#4	#41	#46

Os dados coletados nos experimentos realizados utilizando os valores de parâmetros dados na Tabela 6.6 estão apresentados na Tabela A.1 no Apêndice A. A partir desses dados é construída a análise de tendência de parâmetros que é mostrada na Figura 6.8.

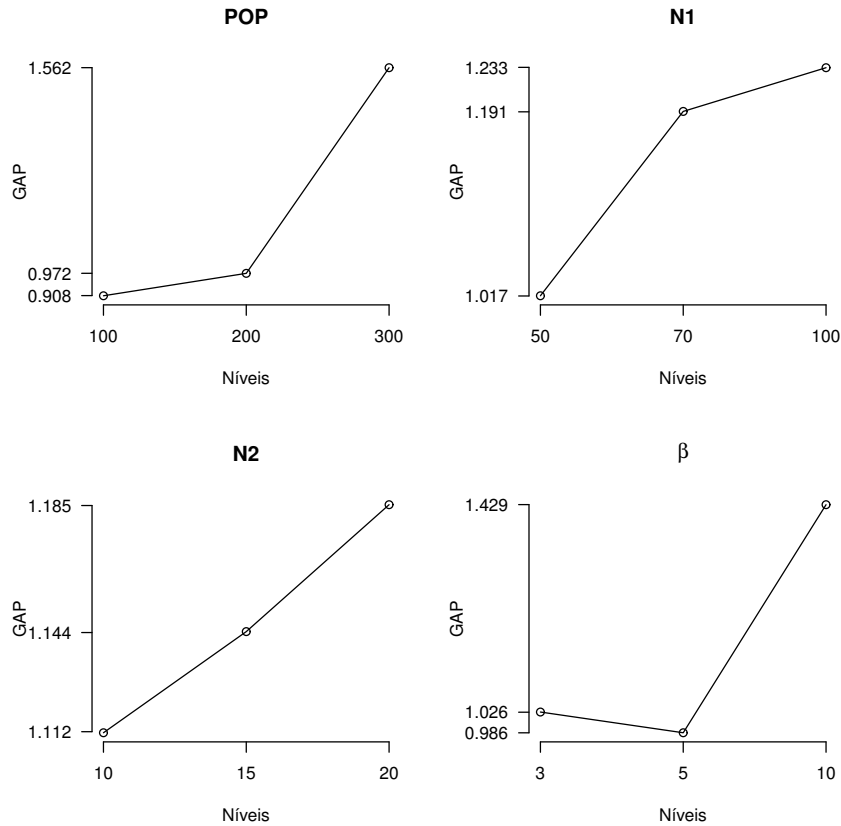


Figura 6.8: Tendência dos parâmetros do CLONALG-MRCPSP ao variar o nível.

Essa tabela mostra a variação de GAP , definido na Equação (6.1), em relação a variação dos valores de parâmetros. Para os parâmetros POP , $N1$ e $N2$, o valor de GAP cresce com o aumento do valor desses parâmetros. Isso significa, para o critério de parada adotado, que valores baixos para esses 3 parâmetros são desejáveis. Em particular, para o parâmetro β , quando ocorre a variação de 3 para 5, o valor de GAP diminuiu. Porém, quando há a variação de 5 para 10, o valor de GAP aumentou. Como já comentado, esse parâmetro fornece a quantidade de clones por solução para explorar a vizinhança de cada solução, por meio dos movimentos $N1$, $N2$ e $N3$. Então, o nível 5 indica uma possível escolha para o parâmetro β . Para todos esses 4 parâmetros, se houver aumento do nível, o CLONALG-MRCPSP possui uma tendência de executar menor número de gerações, devido ao critério de parada. A influência relativa desses parâmetros é identificada pela Tabela 6.7.

Tabela 6.7: Respostas entre as médias das combinações de níveis de parâmetros

Nível	POP	N1	N2	β
1	0.9075019	1.017189	1.111656	1.026330
2	0.9718593	1.191370	1.144306	0.986087
3	1.5618667	1.232669	1.185267	1.428811
Δ	0.6543648	0.2154796	0.07361111	0.4427241
Rank	1	3	4	2

As melhores respostas, em média, estão destacadas em negrito na Tabela 6.7. De acordo com o valor Δ , percebe-se que os parâmetros que mais afetam a convergência da metaheurística CLONALG-MRCPSP são, respectivamente, POP e β . O parâmetro POP é responsável por explorar o espaço de soluções, enquanto β realiza intensificações das soluções por meio da mutação. Observa-se que o parâmetro $N2$, responsável por manter a diversificação das soluções a cada geração, possui, no contexto da análise realizada, menor influência no algoritmo. Uma possível justificativa para essa última análise é devido a mutação modificar, no máximo, 50% de uma solução, de acordo com a discussão na definição do Algoritmo 9. Então, a mutação também acrescenta influência na diversificação, possibilitando valores baixos para o nível de $N2$. A Figura 6.9 apresenta boxplots para cada parâmetro.

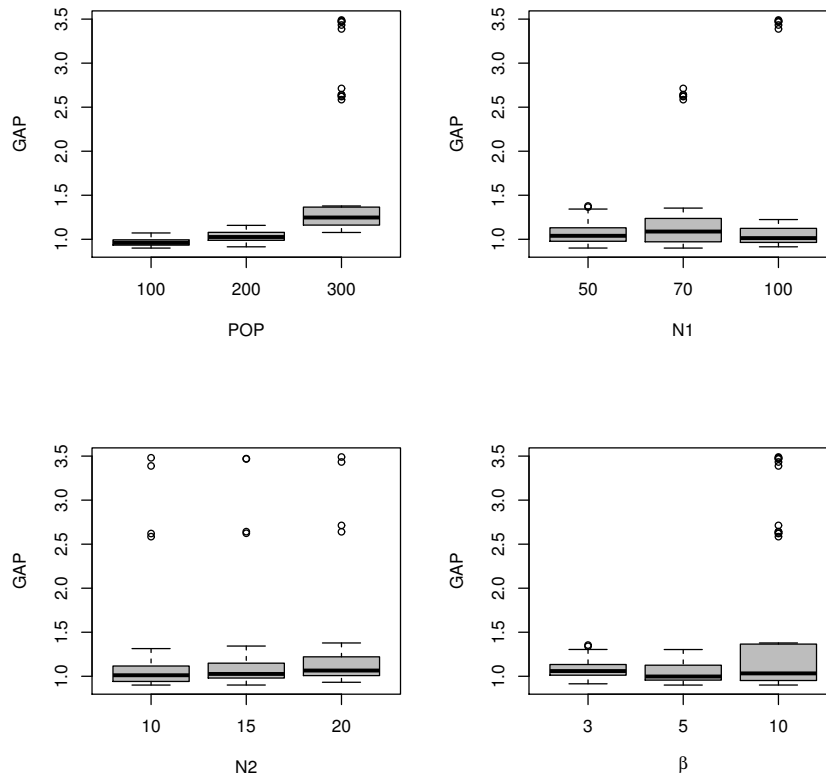


Figura 6.9: Boxplot para os níveis de cada parâmetro.

Para o parâmetro POP , o menor valor apresentou menor mediana, assim como menor variação de valores de GAP . Identifica-se que o valor 300 apresentou *outliers*, ou seja, esse nível apresentou alguns valores de GAP distantes da maioria das observações. O parâmetro $N2$ apresenta variações semelhantes entre os níveis com a menor mediana para o menor valor, sendo esse último igual a 10. Todos os valores apresentam *outliers*. O parâmetro $N1$ apresentou a menor mediana no nível 100. O nível 70 foi aquele que apresentou maior variância entre as respostas e todos os níveis apresentaram *outliers*, porém, o nível 50 apresentou *outliers* próximos da maioria das soluções. Quando verificou-se a média das respostas, pela Tabela 6.7, o nível 50 apresentou menor valor, então os *outliers* influenciaram, em média, as respostas

entre os níveis 50 e 100 do parâmetro $N1$. Para $\beta = 5$, foi obtida a menor mediana com ausência de *outliers*. Destaca-se que o nível $\beta = 3$ apresentou a menor variação entre as respostas com *outliers* próximos da maioria das observações. Para todos os parâmetros, houve sobreposição entre as caixas, não permitindo afirmar pelo box-plot qual o nível possui a melhor resposta. A Figura 6.10 apresenta os gráficos de interação entre os pares de parâmetros.

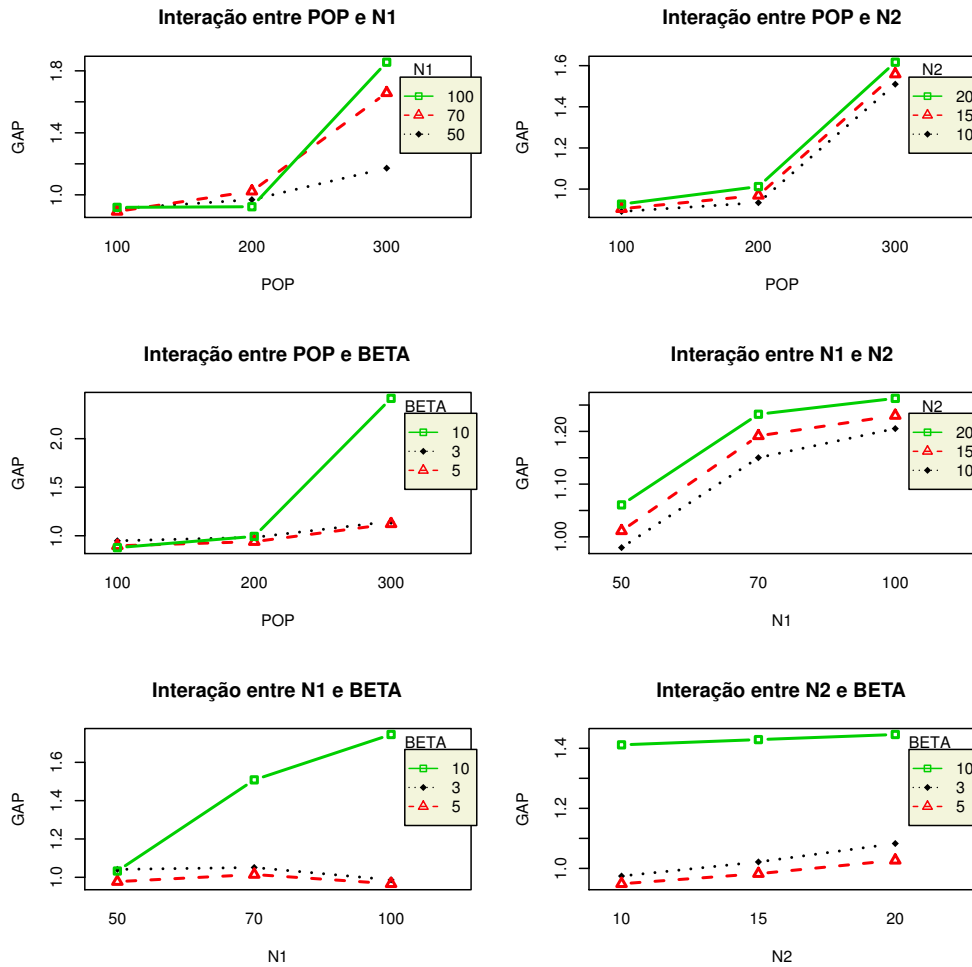


Figura 6.10: Interação entre os parâmetros POP , $N1$, $N2$ e β por meio da variação de cada nível.

Por meio das observações de cada nível, não é possível afirmar que existe interação entre os pares $(POP, N2)$, $(N1, N2)$, $(N1, \beta)$ e $(N2, \beta)$, já que as retas não se cruzam para os níveis analisados. Por esses gráficos, fica claro a interação entre $(POP, N1)$ e (POP, β) . A Tabela 6.8 apresenta o resultado do teste ANOVA sobre os parâmetros em análise.

Tabela 6.8: Tabela ANOVA para $\alpha = 0.05$

	DF	SS	MS	F_0	p_{valor}
<i>POP</i>	2	0.22023	0.11012	18342.090	2^{-16}
<i>N1</i>	2	0.01366	0.00683	1138.010	2^{-16}
<i>N2</i>	2	0.00396	0.00198	329.610	2^{-16}
β	2	0.07025	0.03512	5850.802	2^{-16}
<i>POP</i> : <i>N1</i>	4	0.03108	0.00777	1294.267	2^{-16}
<i>POP</i> : <i>N2</i>	4	0.00042	0.00011	17.639	1.9^{-10}
<i>N1</i> : <i>N2</i>	4	0.00010	0.00002	4.030	0.00496
<i>POP</i> : β	4	0.14730	0.03682	6133.801	2^{-16}
<i>N1</i> : β	4	0.03372	0.00843	1404.284	2^{-16}
<i>N2</i> : β	4	0.00083	0.00021	34.609	2^{-16}
<i>POP</i> : <i>N1</i> : <i>N2</i>	8	0.00002	0.00000	0.418	0.90725
<i>POP</i> : <i>N1</i> : β	8	0.05271	0.00659	1097.574	2^{-16}
<i>POP</i> : <i>N2</i> : β	8	0.00013	0.00002	2.728	0.01028
<i>N1</i> : <i>N2</i> : β	8	0.00003	0.00000	0.587	0.78614
<i>POP</i> : <i>N1</i> : <i>N2</i> : β	16	0.00012	0.00001	1.237	0.25939
Resíduos	81	0.00049	0.00001		

Os valores p_{value} destacados em negrito na Tabela 6.8 informam a diferença estatística entre as médias de combinação de parâmetros, com 95% de confiança. Observa-se diferenças entre as médias dos parâmetros das combinações de ordem 2 e de ordem 3 para $(POP, N1, \beta)$ e $(POP, N2, \beta)$. Boxplots para essas combinações de ordem 2 e 3 são apresentados no Apêndice B. A análise dos boxplots apresenta influência significativa do parâmetro *POP*, quando combinado com outros parâmetros. O valor *POP* = 100 realmente apresenta os menores valores de *GAP*. Observou-se que os níveis $\beta = 5$ e $\beta = 10$ apresentam menores valores de *GAP*, principalmente quando combinados com *POP* = 100. Já para os parâmetros *N1* e *N2* não é possível afirmar quais níveis destacam-se na obtenção dos menores valores de *GAP*. Essa análise corrobora o resultado discutido na Tabela 6.7, exceto para o parâmetro β . Entretanto, para verificar diferença estatística significativa, foi necessário realizar o teste de Tukey para todas as combinações de parâmetros destacadas na Tabela 6.8. Os testes Shapiro-Wilk, Bartlett e Durbin-Watson apresentam valores p_{value} , respectivamente, 0.1878, 0.9617 e 0.056, para $\alpha = 0.05$. Esses valores não negam as hipóteses H_0 , conforme apresentadas na seção 6.1, e portanto, validam o teste ANOVA. A Figura 6.11 apresenta o teste de Tukey para verificação das melhores respostas de cada parâmetro, sendo que as comparações destacadas em vermelho apresentam diferenças estatísticas.

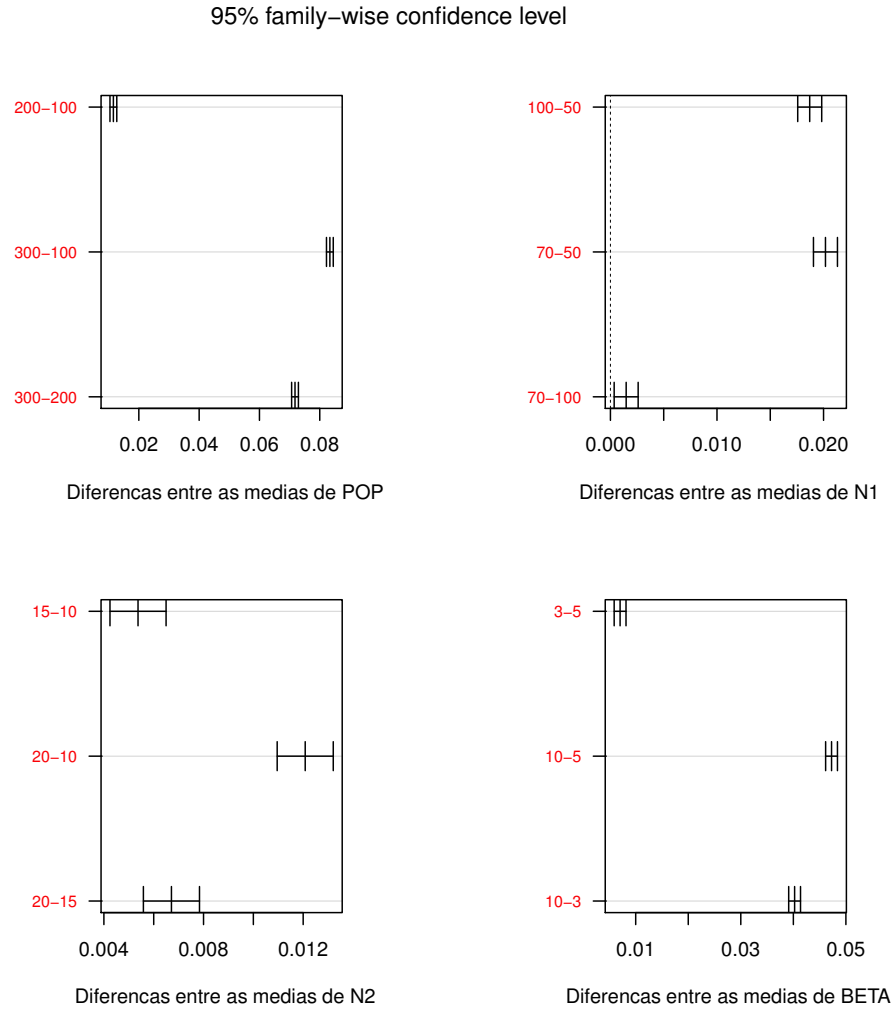


Figura 6.11: Diferença entre as médias dos níveis de cada parâmetro POP , $N1$, $N2$ e β

Analisando o parâmetro POP , existe diferença entre todos os níveis, sendo a maior diferença entre os níveis 300-100. Então, pelas discussões anteriores, o nível 100, de fato, apresenta a melhor resposta para o parâmetro POP . Já o parâmetro $N1$ também apresenta diferenças entre todas as comparações, com destaque para o nível 50. O parâmetro $N2$ apresentou diferenças entre todos os níveis, com destaque para o nível 10. Nota-se, para esse parâmetro, que os intervalos de confiança possuem menores magnitudes em relação aos outros parâmetros, indicando menor diferença entre as variâncias, conforme observado pelo boxplot. Finalmente, o parâmetro β apresenta diferença entre todos os níveis, com destaque para os níveis 3 e 5, com menor diferença entre as médias desses dois níveis. O teste de Tukey para as combinações de ordem 2, que apresentaram diferença estatística, conforme a ANOVA, é apresentado no Apêndice C. Não são apresentados o teste de Tukey para as combinações de ordem 3, que apresentaram diferença estatística, porque os gráficos gerados possuem baixa qualidade de análise.

Ao analisar o teste de Tukey e os boxplots, conclui-se que níveis baixos dos valores de parâmetros analisados tendem a apresentar as melhores respostas, uma vez que o algoritmo tende a executar em maior número de gerações, ao contrário de valores

de níveis altos. Ou seja, é melhor manter uma população com uma quantidade baixa de soluções para que o algoritmo possa refinar essas soluções ao longo de um número maior de gerações. Há a necessidade de manter a diversificação das soluções, porém, nota-se, que valores de níveis baixos apresentaram leve destaque em relação ao valor de nível alto. Sobre o número de clones, β , para cada solução escolhida para clonagem, embora um número maior de clones possa permitir explorar melhor a vizinhança de uma solução, também possui influência na quantidade de gerações do algoritmo CLONALG-MRCPSP. Identificou-se destaque, para os níveis alto e médio do parâmetro β , na determinação dos melhores valores de GAP , embora a tendência desse parâmetro aponta que o nível baixo, em média, apresenta melhores respostas. Isso significa que esse parâmetro possui relação com os outros parâmetros, com maior destaque para a relação $POP : \beta$. Quando POP está em seu nível baixo, os níveis médio e alto de β apresentam boas respostas.

6.4 Comparação entre ILS-MRCPSP e CLONALG-MRCPSP

Ao realizar os testes ANOVA e Tukey nos itens de seções 6.2 e 6.3, o objetivo nesta dissertação para a análise de parâmetros não é informar qual é a melhor combinação de parâmetros que deve ser utilizada, mas apresentar a influência dos parâmetros e a tendência dos valores, assim como das combinações entre valores de parâmetros.

Sendo assim, após as discussões para as análises dos parâmetros, os algoritmos propostos foram comparados utilizando a combinação de parâmetros que apresentou a melhor resposta. Então, os parâmetros foram ajustados como:

- ILS-MRCPSP: $K = 10$, $MAX_MOD = 0.3$.
- CLONALG-MRCPSP: $POP = 100$, $N1 = 70$, $N2 = 10$, $\beta = 5$.

Para essa comparação, todas as classes da PSPLIB foram resolvidas, obtendo então, uma observação GAP_{avr} referente a cada classe. Esse valor indica o GAP esperado para a resolução das respectivas classes pelos algoritmos propostos. A Tabela 6.9 apresenta os resultados obtidos por ambos os algoritmos. Já a Figura 6.12 apresenta o gráfico para essa última tabela. Nessa tabela a coluna “Classe” indica o grupo de instâncias resolvidas, a coluna “Algoritmo” indica as versões propostas dos algoritmos ILS e CLONALG, a coluna “Ótimos” indica a quantidade de soluções ótimas encontradas ou melhores soluções conhecidas para a classe J30, a coluna “ GAP_{avr} ” indica o desvio médio das soluções encontradas, a coluna “Tempo” apresenta os tempos em segundo (e em média) para cada classe resolvida e a coluna “Viabilidade” apresenta a porcentagem de soluções viáveis encontradas para cada classe.

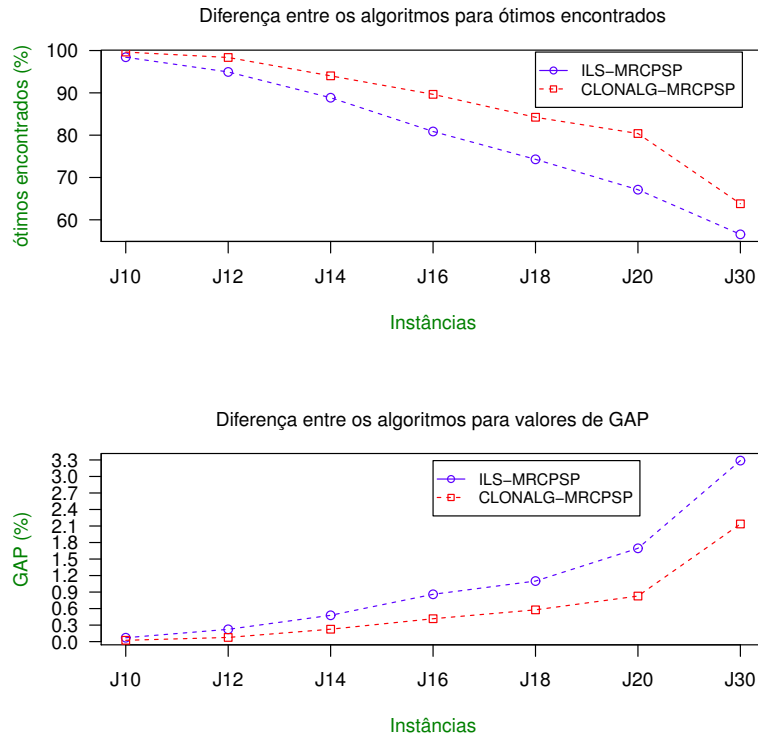


Figura 6.12: Diferença entre os algoritmos para determinação das melhores soluções conhecidas e GAP por instância solucionada.

Tabela 6.9: Comparação entre CLONALG-MRCPSP e ILS-MRCPSP para 5000 soluções geradas

Classe	Algoritmo	Ótimos (%)	GAP_{avr} (%)	Tempo (s)	Viabilidade (%)
J10	ILS-MRCPSP	98.42	0.0705	0.0631	100
	CLONALG-MRCPSP	99.69	0.0242	0.1270	100
J12	ILS-MRCPSP	94.94	0.2231	0.0761	100
	CLONALG-MRCPSP	98.36	0.0758	0.1481	100
J14	ILS-MRCPSP	88.85	0.4779	0.0874	100
	CLONALG-MRCPSP	94.02	0.2253	0.1747	100
J16	ILS-MRCPSP	80.87	0.8594	0.0984	100
	CLONALG-MRCPSP	89.65	0.4161	0.1998	100
J18	ILS-MRCPSP	74.29	1.1001	0.1118	100
	CLONALG-MRCPSP	84.23	0.5770	0.2262	100
J20	ILS-MRCPSP	67.12	1.6949	0.1238	100
	CLONALG-MRCPSP	80.39	0.8276	0.2478	100
J30	ILS-MRCPSP	56.56	3.2882	0.1890	100
	CLONALG-MRCPSP	63.80	2.1368	0.3667	100

Pelas Tabela 6.9 e pela Figura 6.12, o algoritmo CLONALG-MRCPSP supera o algoritmo ILS-MRCPSP para determinar os melhores valores conhecidos, como

também encontra os melhores valores de GAP . Ou seja, o algoritmo CLONALG-MRCPSP resolve as instâncias classes da PSPLIB e apresenta melhores sequenciamentos que o algoritmo ILS-MRCPSP. Porém, como é apresentado na coluna “Tempo” da Tabela 6.9, percebe-se que o algoritmo ILS-MRCPSP exigiu menor tempo computacional. Os boxplots mostrados nas Figuras 6.13, 6.14 e 6.15 confirmam que, o algoritmo CLONALG-MRCPSP encontra uma maior parcela de ótimos e apresenta uma maior parcela de melhores valores de GAP em relação ao algoritmo ILS-MRCPSP, porém o algoritmo ILS-MRCPSP realmente exigiu menor tempo computacional.

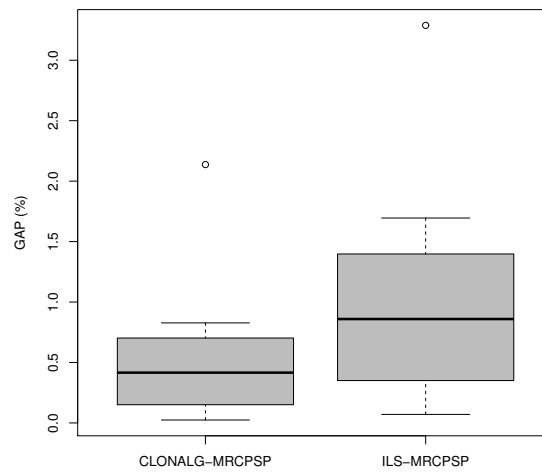


Figura 6.13: Boxplot para a diferença entre os resultados de GAP obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.

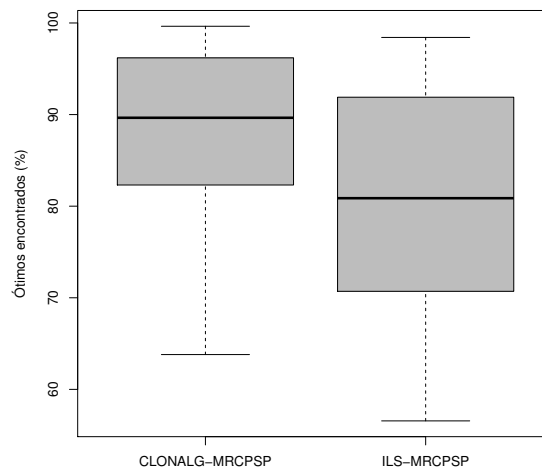


Figura 6.14: Boxplot para a diferença entre os ótimos encontrados obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.

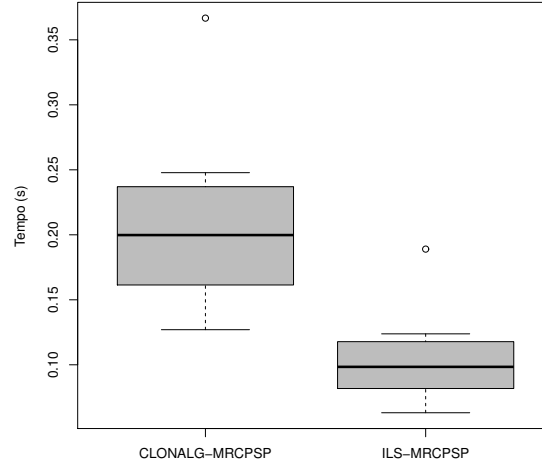


Figura 6.15: Boxplot para a diferença entre os tempos (segundos) obtidos pelos algoritmos ILS-MRCPSP e CLONALG-MRCPSP.

Pelos boxplots 6.13 e 6.14, também é verificado que o algoritmo ILS-MRCPSP apresenta maior variância do que o algoritmo CLONALG-MRCPSP na determinação dos ótimos e valores de GAP . Já em relação ao tempo, o ILS-MRCPSP apresenta menor variância do que o CLONALG-MRCPSP.

Após essas discussões, as vantagens de cada algoritmo são apresentadas. Em relação à metaheurística ILS-MRCPSP, esse algoritmo possui simplicidade de codificação, poucos parâmetros para ajuste e baixo tempo computacional para execução. Já o algoritmo CLONALG-MRCPSP apresentou as melhores soluções por meio da maior parcela de ótimos e menores valores de GAP . Mesmo exigindo maior tempo computacional em relação ao ILS-MRCPSP, o algoritmo CLONALG-MRCPSP demandou tempo computacional aceitável para convergência. É importante evidenciar a capacidade de ambos os algoritmos finalizarem a execução com 100% de soluções viáveis.

6.5 Comparação com a literatura

Para comparação com os trabalhos presentes na literatura, os resultados obtidos pela metaheurística CLONALG-MRCPSP foram utilizados. Os trabalhos citados nessa seção apresentam os resultados mais influentes presentes na literatura, para os últimos 10 anos. As Tabelas 6.10 e 6.11 apresentam os resultados, respectivamente, para GAP_{avr} e ótimos encontrados. Para a classe J30, em relação à Tabela 6.10, o GAP_{avr} é apresentado como desvio médio para o limite inferior conhecido. Já em relação à Tabela 6.11 para a classe J30, o resultado é apresentado como obtenção das melhores soluções conhecidas.

Tabela 6.10: Valores de GAP_{avr} (%) obtidos para as classes da PSPLIB.

Autor	Classe						
	J10	J12	J14	J16	J18	J20	J30
Zhang et al. (2006)*	0.11	0.17	0.41	0.83	1.33	1.79	-
Jarboui et al. (2008)	0.03	0.09	0.36	0.44	0.89	1.10	18.14
Chiang et al. (2008)	0.15	-	-	-	-	2.42	15.18
Lova et al. (2009)**	0.06	0.17	0.32	0.44	0.63	0.87	14.77
Tseng e Chen (2009)	0.33	0.524	0.917	1.087	1.301	1.713	18.332
Damak et al. (2009)	0.74	-	-	-	-	1.62	15.43
Van Peteghem e Vanhoucke (2010)	0.01	0.09	0.22	0.32	0.42	0.57	13.75
Elloumi e Fortemps (2010)**	0.14	0.24	0.77	0.91	1.30	1.62	16.16
Wauters et al. (2011)	0.05	0.08	0.23	0.30	0.53	0.70	14.62
Muller (2011)	0.05	0.20	0.55	0.78	1.14	1.52	15.96
Van Peteghem e Vanhoucke (2011)	0.00	0.02	0.08	0.15	0.23	0.32	13.66
Coelho e Vanhoucke (2011)	0.07	0.16	0.32	0.48	0.56	0.80	14.44
Wang e Fang (2011)	0.103	0.21	0.462	0.578	0.944	1.399	13.461
Wang e Fang (2012)	0.12	0.14	0.43	0.59	0.90	1.28	-
Li e Zhang (2013)	0.09	0.13	0.40	0.57	1.02	1.10	-
Soliman e Elgendi (2014b)	0.08	0.14	0.22	0.40	0.72	1.08	-
Soliman e Elgendi (2014a)**	0.094	0.12	0.361	0.424	0.851	1.093	12.704
CLONALG-MRCPSP	0.0242	0.0758	0.2253	0.4161	0.5770	0.8276	14.1632

(*) Nesse trabalho o critério de parada foi dado como 6000 sequenciamentos gerados.

(**) O algoritmo terminou com solução inviável em alguma execução para a classe J30.

Por essa tabela, claramente, as heurísticas de Van Peteghem e Vanhoucke (2010) e Van Peteghem e Vanhoucke (2011) destacam-se na obtenção dos menores valores de GAP , em média. O algoritmo CLONALG-MRCPSP obtém resultados semelhantes a Lova et al. (2009), Wauters et al. (2011) e Coelho e Vanhoucke (2011), sendo que esses trabalhos recebem destaque na literatura, conforme Van Peteghem e Vanhoucke (2014) e Mika et al.). Soliman e Elgendi (2014a) apresentam boa resolução para a classe J30, todavia os autores destacam que não obtiveram uma taxa de 100% de soluções viáveis ao solucionar essa classe.

Tabela 6.11: Ótimos (%) obtidos para as classes da PSPLIB.

Autor	Classe						
	J10	J12	J14	J16	J18	J20	J30
Zhang et al. (2006)*	97.9	95.2	89.3	85.6	74.5	69.1	-
Jarboui et al. (2008)	99.25	98.47	91.11	85.91	79.89	74.19	57.41
Lova et al. (2009)**	98.51	96.53	92.92	90.00	84.96	80.32	52.35
Tseng e Chen (2009)	95.156	90.567	82.033	77.388	73.376	66.655	-
Van Peteghem e Vanhoucke (2010)	99.63	98.17	94.56	92.00	88.95	85.74	71.0
Elloumi e Fortemps (2010)**	97.31	94.94	83.84	80.69	72.92	67.43	-
Wauters et al. (2011)	98.70	98.17	94.74	92.18	86.23	81.59	-
Muller (2011)	99.01	95.59	88.75	84.76	78.75	72.87	62.97
Van Peteghem e Vanhoucke (2011)	100	99.45	97.28	96.73	94.75	87.73	-
Wang e Fang (2011)	97.929	95.978	90.862	86.491	79.438	72.84	-
Li e Zhang (2013)	98.3	96.5	90.3	86.9	76.1	72.4	43.8
Soliman e Elgendi (2014a)**	97.629	97.111	92.313	91.059	82.461	77.575	-
Okada et al. (2014)	99.43	98.54	95.95	93.22	90.22	85.99	69.82
CLONALG-MRCPSP	99.69	98.36	94.02	89.65	84.23	80.39	63.80

(*) Nesse trabalho o critério de parada foi dado como 6000 sequenciamentos gerados.

(**) O algoritmo terminou com solução inviável em alguma execução para a classe J30.

Novamente, pela Tabela 6.11, as heurísticas de Van Peteghem e Vanhoucke (2010) e Van Peteghem e Vanhoucke (2011) destacam-se na obtenção da maior quantidade de ótimos determinados, em adição com a heurística de Okada et al. (2014). O

algoritmo CLONALG-MRCPSP obtém resultados semelhantes aos encontrados em [Lova et al. \(2009\)](#) e [Wauters et al. \(2011\)](#), superando as outras heurísticas propostas.

Pelas Tabelas 6.10 e 6.11, conclui-se que o algoritmo CLONALG-MRCPSP é capaz de gerar soluções competitivas em relação às soluções geradas por vários procedimentos no estado-da-arte. Além disso destaca-se que o algoritmo CLONALG-MRCPSP pode ser facilmente adaptado para solucionar extensões do MRCPSP, sejam essas extensões relacionadas a alterações na função objetivo ou no consumo de recursos.

Capítulo 7

Conclusões

O sequenciamento em projetos é aplicado em diversos contextos para situações reais. Assim, os algoritmos que oferecem boas respostas a projetos que apresentam diversas formas de executar uma tarefa ganham destaque como boas ferramentas de apoio à decisão. Nessa dissertação foi tratado o problema de sequenciamento em projetos com restrição em recursos e múltiplos modos de execução, com o objetivo de minimizar o *makespan* (MRCPSP). De acordo com a complexidade do problema, foi verificado que várias técnicas exatas e heurísticas são propostas para resolução do MRCPSP. Nos últimos anos, as técnicas heurísticas receberam grande atenção por parte dos pesquisadores, pois oferecerem soluções próximas do valor ótimo em tempo computacional inferior em relação aos métodos exatos. Então, nesta dissertação foram propostas as implementações das metaheurísticas *Iterated Local Search* (ILS-MRCPSP) e Seleção Clonal (CLONALG-MRCPSP) baseada em Sistemas Imunológicos Artificiais para a solução do MRCPSP.

Por meio da literatura relacionada ao problema discutido nesta dissertação, foram identificadas técnicas heurísticas que recebem destaque na resolução do MRCPSP, como Regras de Prioridade, o esquema de sequenciamento Serial e a busca local *Multi-mode forward/backward improvement*. Então, as metaheurísticas implementadas nesta dissertação, com adição dessas técnicas heurísticas como tentativa de explorar as características do problema em discussão, apresentaram boa capacidade de resolução da biblioteca PSPLIB. Testes estatísticos foram realizados para identificar tendências dos parâmetros, bem como realizar comparações entre os algoritmos propostos. De acordo com os experimentos realizados, o destaque para o algoritmo ILS-MRCPSP foi o tempo computacional exigido, enquanto o algoritmo CLONALG-MRCPSP apresentou melhores soluções para as classes testadas. Em relação aos resultados presentes na literatura, o algoritmo CLONALG-MRCPSP apresenta resultados competitivos em comparação com os resultados mais influentes na literatura.

Como propostas futuras, outras regras de prioridade podem ser exploradas para a aplicação dessas regras tanto na fase de construção, quanto na fase de modificação das soluções. Também, outros tipos de movimentos podem ser explorados, como foi proposto em [Araujo et al. \(2016\)](#). Em relação ao algoritmo ILS, um histórico pode ser utilizado para evitar gerar soluções já verificadas em um período recente, como são utilizadas as listas tabu na Busca Tabu. Já para o algoritmo Seleção Clonal, técnicas que permitem manter soluções com grau de diferença entre si podem

ser acrescentadas a essa metaheurística, com o objetivo de manter vários ótimos locais durante toda a busca do algoritmo. Também, as técnicas conhecidas como *matheuristics*, que são heurísticas hibridizadas com programação exata, podem ser utilizadas. [Toffolo et al. \(2013\)](#) aplicaram essa estratégia para a resolução de uma variação do MRCPSP.

Ao longo do desenvolvimento desta dissertação, foram apresentados em congressos acadêmicos os seguintes artigos:

- *Uma abordagem baseada em iterated local search para resolver o problema de escalonamento de projetos com restrição de recursos e múltiplos modos de execução.* XVIII Simpósio de Pesquisa Operacional & Logística da Marinha, SPOLM 2015;
- *Uma abordagem baseada em iterated local search com multistart para resolver o problema de escalonamento de projetos com restrição de recursos e múltiplos modos de execução.* XLVII Simpósio Brasileiro de Pesquisa Operacional, SBPO 2015;
- *Algoritmo híbrido de Seleção Clonal Baseado em Regras de Prioridade para o Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução.* XXXVI Iberian Latin American Congress on Computational Methods in Engineering, CILAMCE 2015;
- *A clonal selection algorithm with local search for solving the multimode resource-constrained project scheduling problem.* Workshop on Applied Combinatorial Optimization Methods, WACOM 2016.

Referências Bibliográficas

Afshar-Nadjafi, Behrouz. (2014). Multi-mode resource availability cost problem with recruitment and release dates for resources. *Applied Mathematical Modelling*, v. 38, n. 21, p. 5347–5355.

Agarwal, Rina; Tiwari, MK e Mukherjee, SK. (2007). Artificial immune system based approach for solving resource constraint project scheduling problem. *The International Journal of Advanced Manufacturing Technology*, v. 34, n. 5-6, p. 584–593.

Agarwal, Shivam e Bhat, Tushar. (2013). Evolutionary algorithm for solving multi-mode resource constrained project scheduling problems through deterministic mode selection. *International Journal of Computer Applications*, v. 78, n. 13, p. 14–19.

Alcaraz, Javier; Maroto, Concepcion e Ruiz, Ruben. (2003). Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *Journal of the Operational Research Society*, v. 54, n. 6, p. 614–626.

Andreica, Anca e Chira, Camelia. (2014). Best-order crossover in an evolutionary approach to multi-mode resource-constrained project scheduling. *International Journal of Computer Information Systems and Industrial Management (IJCISIM)*, v. 6, p. 364–372.

Araujo, Janniele AS; Santos, Haroldo G; Baltar, Davi D; Toffolo, Túlio AM e Wauters, Tony. (2016). Neighborhood composition strategies in stochastic local search. *Proceedings of the International Workshop on Hybrid Metaheuristics*, p. 118–130. Springer, (2016).

Atli, Omer e Kahraman, Cengiz. (2012). Multi mode resource constrained project scheduling problems with solving taboo search. Kahraman, Cengiz; Bozbura, Faik Tunc e Kerre, Etienne E, editors, *Uncertainty Modeling in Knowledge Engineering and Decision Making*, volume 7, p. 448–453. World Scientific.

Ballestin, Francisco e Trautmann, Norbert. (2008). An iterated local search heuristic for the resource-constrained weighted earliness-tardiness project scheduling problem. *International Journal of Production Research*, v. 46, n. 22, p. 6231–6249.

Bedworth, David D. e Bailey, James E. (1982). *Integrated Production, Control Systems: Management, Analysis, and Design*. John Wiley & Sons, Inc., New York, NY, USA.

- Bilolikar, V; Jain, Karuna e Sharma, M. (2012). An annealed genetic algorithm for multi mode resource constrained project scheduling problem. *International Journal of Computers and Applications*, v. 60, n. 1, p. 36–42.
- Bilolikar, Vijay S. (2014). An adaptive crossover genetic algorithm for multi-mode resource constrained project scheduling with discounted cash flows. Narayanan, Sri-ram e Bhageria, Rishabh, editors, *Proceedings of the POMS 26th Annual Conference on Production and Operations Management Society*, p. 1–9, (2014).
- Błażewicz, Jacek. (1986). *Scheduling under resource constraints: Deterministic models*, volume 7. JC Baltzer.
- Blazewicz, Jacek; Lenstra, Jan Karel e Kan, AHG Rinnooy. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, v. 5, n. 1, p. 11–24.
- Boctor, Fayez F. (1996). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research*, v. 90, n. 2, p. 349–361.
- Boctor, Fayez Fouad. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *The International Journal of Production Research*, v. 31, n. 11, p. 2547–2558.
- Bouleimen, Klein e Lecocq, Housini. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research*, v. 149, n. 2, p. 268–281.
- Browning, Tyson R e Yassine, Ali A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, v. 126, n. 2, p. 212–228.
- Brucker, Peter; Drexl, Andreas; Mohring, Rolf; Neumann, Klaus e Pesch, Erwin. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, v. 112, n. 1, p. 3–41.
- Buddhakulsomsiri, Jirachai e Kim, David S. (2007). Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, v. 178, n. 2, p. 374–390.
- Cheng, Min-Yuan e Tran, Duc-Hoc. (2016). An efficient hybrid differential evolution based serial method for multimode resource-constrained project scheduling. *KSCE Journal of Civil Engineering*, v. 20, n. 1, p. 90–100.
- Chiang, Chuan-Wen; Huang, Yu-Qing e Wang, Wen-Yen. (2008). Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, v. 19, n. 4, 5, p. 345–358.

- Coelho, José e Vanhoucke, Mario. (2011). Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. *European Journal of Operational Research*, v. 213, n. 1, p. 73–82.
- Coello, Carlos A Coello; Rivera, Daniel Cortes e Cortes, Nareli Cruz. (2003). Use of an artificial immune system for job shop scheduling. *In Artificial Immune Systems*, p. 1–10. Springer.
- Colak, Selcuk; Agarwal, Anurag e Erenguc, Selcuk. (2013). Multi-mode resource-constrained project-scheduling problem with renewable resources: New solution approaches. *Journal of Business & Economics Research (JBER)*, v. 11, n. 11, p. 455–468.
- Cui, Jianshuang e Yu, Liruoyang. (2014). An efficient discrete particle swarm optimization for solving multi-mode resource-constrained project scheduling problem. *Proceedings of the 2014 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, p. 858–862. IEEE, (2014).
- Damak, N; Jarboui, Bassem; Siarry, Patrick e Loukil, Taicir. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research*, v. 36, n. 9, p. 2653–2659.
- De Castro, Leandro N e Von Zuben, Fernando J. (2002). Learning and optimization using the clonal selection principle. *Evolutionary Computation on IEEE Transactions*, v. 6, n. 3, p. 239–251.
- De Castro, Leandro Nunes. (2006). *Fundamentals of natural computing: basic concepts, algorithms, and applications*. Chapman & Hall / CRC Press.
- Demeulemeester, Erik L. (2002). *Project scheduling: a research handbook*, volume 102. Springer.
- Diana, Rodney Oliveira Marinho; França Filho, Moacir Felizardo de; Souza, Sérgio Ricardo de e Almeida Vitor, João Francisco de. (2015). An immune-inspired algorithm for an unrelated parallel machines scheduling problem with sequence and machine dependent setup-times for makespan minimisation. *Neurocomputing*, v. 163, p. 94–105.
- Drexl, Andreas. (1991). Scheduling of project networks by job assignment. *Management Science*, v. 37, n. 12, p. 1590–1602.
- Drexl, Andreas e Gruenewald, Juergen. (1993). Nonpreemptive multi-mode resource-constrained project scheduling. *IIE transactions*, v. 25, n. 5, p. 74–81.
- Elloumi, Sonda e Fortemps, Philippe. (2010). A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 205, n. 1, p. 31–41.
- Elmaghraby, Salah E. (1977). *Activity networks: Project planning and control by network models*. Wiley New York.

- Engin, Orhan e Döyen, Alper. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, v. 20, n. 6, p. 1083–1095.
- Glover, Fred e Kochenberger, Gary A. (2003). *Handbook of metaheuristics*. Springer Science & Business Media.
- Hart, Emma; Ross, Peter e Nelson, Jeremy. (1998). Producing robust schedules via an artificial immune system. *Proceedings of the Evolutionary Computation, 1998. IEEE World Congress on Computational Intelligence*, p. 464–469. IEEE, (1998).
- Hartmann, Sonke. (2001). Project scheduling with multiple modes: a genetic algorithm. *Annals of Operations Research*, v. 102, n. 1-4, p. 111–135.
- Hartmann, Sönke e Briskorn, Dirk. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 207, n. 1, p. 1–14.
- Hartmann, Sönke e Drexel, Andreas. (1998). Project scheduling with multiple modes: a comparison of exact algorithms. *Networks*, v. 32, n. 4, p. 283–297.
- Hartmann, Sönke e Kolisch, Rainer. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 127, n. 2, p. 394–407.
- Hu, Shicheng; Zhang, Zhaoze; He, Qingsong e Sun, Xuedong. (2013). An iterated local search algorithm for a place scheduling problem. *Mathematical Problems in Engineering*, v. 2013, p. 7.
- J. Durbin, G. S. Watson. (1950). Testing for serial correlation in least squares regression: I. *Biometrika*, v. 37, n. 3/4, p. 409–428.
- Jarboui, Bassem; Damak, N; Siarry, Patrick e Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation*, v. 195, n. 1, p. 299–308.
- Jedrzejowicz, Piotr e Ratajczak, Ewa. (2006). Population learning algorithm for the resource-constrained project scheduling. *Perspectives in Modern Project Scheduling*, v. 92, p. 275–296.
- Jedrzejowicz, Piotr e Ratajczak-Ropel, Ewa. (2007). Agent-based approach to solving the resource constrained project scheduling problem. *In Adaptive and Natural Computing Algorithms*, p. 480–487. Springer.
- Jozefowska, Joanna; Mika, Marek; Rozycki, Rafal; Waligora, Grzegorz e Weglarz, Jan. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, v. 102, n. 1-4, p. 137–155.
- Ju, Chunhua e Chen, Tinggui. (2012). Simplifying multiproject scheduling problem based on design structure matrix and its solution by an improved aiNet algorithm. *Discrete Dynamics in Nature and Society*, v. 2012, p. 22.

Jünger, Michael; Liebling, Thomas M; Naddef, Denis; Nemhauser, George L; Puleyblank, William R; Reinelt, Gerhard; Rinaldi, Giovanni e Wolsey, Laurence A. (2009). *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*. Springer Science & Business Media.

Kazemi, FS e Tavakkoli-Moghaddam, R. (2011). Solving a multi-objective multi-mode resource-constrained project scheduling problem with discounted cash flows. *International Journal of Academic Research*, v. 3, n. 1, p. 103–110.

Kelley, J. E. (1963). The critical-path method: Resources planning and scheduling. Muth, J.F. e Thompson, G.L., editors, *In Industrial Scheduling*, p. 347–365. Prentice-Hall.

Kelley Jr, James E e Walker, Morgan R. (1959). Critical-path planning and scheduling. *Papers presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM computer conference*, p. 160–173. ACM, (1959).

Khalilzadeh, Mohammad. (2015). A honey bee swarm optimization algorithm for minimizing the total costs of resources in MRCPSP. *Indian Journal of Science and Technology*, v. 8, n. 11, p. 8.

Khalilzadeh, Mohammad; Kianfar, Fereydoon; Shirzadeh Chaleshtari, Ali; Shadrokh, Shahram e Ranjbar, Mohammad. (2012). A modified PSO algorithm for minimizing the total costs of resources in MRCPSP. *Mathematical Problems in Engineering*, v. 2012, p. 18.

Knotts, Gary; Dror, Moshe e Hartman, Bruce C. (2000). Agent-based project scheduling. *IIE Transactions*, v. 32, n. 5, p. 387–401.

Kolisch, Rainer. (1995). *Project scheduling under resource constraints: efficient heuristics for several problem classes*. Springer-Physica-Verlag Heidelberg.

Kolisch, Rainer. (1996)a. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management*, v. 14, n. 3, p. 179–192.

Kolisch, Rainer. (1996)b. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, v. 90, n. 2, p. 320–333.

Kolisch, Rainer e Drexel, Andreas. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics (NRL)*, v. 43, n. 1, p. 23–40.

Kolisch, Rainer e Drexel, Andreas. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions*, v. 29, n. 11, p. 987–999.

Kolisch, Rainer e Hartmann, Sönke. (1999). Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. *In Project scheduling*, p. 147–178. Springer.

- Kolisch, Rainer e Hartmann, Sonke. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, v. 174, n. 1, p. 23–37.
- Kolisch, Rainer e Sprecher, Arno. (1997). PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, v. 96, n. 1, p. 205–216.
- Kramer, Bradley A e Hwang, Ching-Lai. (1991). Resource constrained project scheduling: Modelling with multiple alternatives. *Mathematical and computer modelling*, v. 15, n. 8, p. 49–63.
- Lawler, Eugene L. (1976). *Combinatorial optimization: networks and matroids*. Courier Corporation.
- Li, Heng e Zhang, Hong. (2013). Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction*, v. 35, p. 431–438.
- Li, KY e Willis, RJ. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research*, v. 56, n. 3, p. 370–379.
- Lombardi, Michele e Milano, Michela. (2012). Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. *Constraints*, v. 17, n. 1, p. 51–85.
- Lorenzoni, Luciano Lessa; Ahonen, Hannu e Alvarenga, Arlindo Gomes de. (2006). A multi-mode resource-constrained scheduling problem in the context of port operations. *Computers & Industrial Engineering*, v. 50, n. 1, p. 55–65.
- Lourenço, Helena R; Martin, Olivier C e Stützle, Thomas. (2003). Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, p. 320–353. Springer.
- Lova, A; Tormos, P e Barber, F. (2006). Multi-mode resource constrained project scheduling: Scheduling schemes, priority rules and mode selection rules. *Inteligencia artificial*, v. 30, n. 10, p. 69–86.
- Lova, Antonio; Tormos, Pilar; Cervantes, Mariamar e Barber, Federico. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics*, v. 117, n. 2, p. 302–316.
- Messelis, Tommy e De Causmaecker, Patrick. (2014). An automatic algorithm selection approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 233, n. 3, p. 511–528.
- Mika, Marek; Waligóra, Grzegorz e Weglarz, Jan.). Overview and state of the art.
- Mika, Marek; Waligóra, Grzegorz e Weglarz, Jan. (2008). Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research*, v. 187, n. 3, p. 1238–1250.

- Mirzaei, Omid e Akbarzadeh-T, Mohammad-R. (2013). A novel learning algorithm based on a multi-agent structure for solving multi-mode resource-constrained project scheduling problem. *Journal of Convergence*, v. 4, n. 1, p. 47–52.
- Montgomery, Douglas C. (2013). *Design and analysis of experiments*, volume 8. John Wiley & Sons, Inc.
- Mori, Masao e Tseng, Ching Chih. (1997). A genetic algorithm for multi-mode resource constrained project scheduling problem. *European Journal of Operational Research*, v. 100, n. 1, p. 134–141.
- Muller, Laurent Flindt. (2011). An adaptive large neighborhood search algorithm for the Multi-mode RCPSP. *Tech. Report 3/2011, Department of Management Engineering, Technical University of Denmark*. DTU Management, online: <http://orbit.dtu.dk/ws/files/5704914/rap3saml.2011.pdf>.
- Nader Abadi, Sedigheh; Roghanian, Emad e Aghassi, Hadi. (2011). A multi-mode resource-constrained optimization of time-cost trade-off problems in project scheduling using a genetic algorithm. *Journal of Optimization in Industrial Engineering*, v. 4, n. 8, p. 55–64.
- Najid, Najib M e Arroub, Marouane. (2010). An efficient algorithm for the multi-mode resource constrained project scheduling problem with resource flexibility. *International Journal of Mathematics in Operational Research*, v. 2, n. 6, p. 748–761.
- Okada, Ikutaro; Takahashi, Koji; Zhang, Wenqiang; Zhang, Xiaofu; Yang, Hongyu e Fujimura, Shigeru. (2014). A genetic algorithm with local search using activity list characteristics for solving resource-constrained project scheduling problem with multiple modes. *IEEE Transactions on Electrical and Electronic Engineering*, v. 9, n. 2, p. 190–199.
- Özdamar, Linet. (1999). A genetic algorithm approach to a general category project scheduling problem. *Systems, Man, and Cybernetics, Part C: Applications and Reviews on IEEE Transactions*, v. 29, n. 1, p. 44–59.
- Özdamar, Linet e Ulusoy, Gündüz. (1994). A local constraint based analysis approach to project scheduling under general resource constraints. *European Journal of Operational Research*, v. 79, n. 2, p. 287–298.
- Özdamar, Linet e Ulusoy, Gündüz. (1996). A note on an iterative forward/backward scheduling technique with reference to a procedure by Li and Willis. *European Journal of Operational Research*, v. 89, n. 2, p. 400–407.
- Papadimitriou, Christos H e Steiglitz, Kenneth. (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.
- Patterson, Slowinski-R. Talbot F.B.J.H. e Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. Slowinski, R. e Weglarz, J., editors, *In Advances in Project Scheduling*, p. 3–28. Elsevier.

- Phruksaphanrat, B. (2014). Multi-objective multi-mode resource-constrained project scheduling problem by preemptive fuzzy goal programming. *World Academy of Science, Engineering and Technology, International Journal of Mechanical, Aerospace, Industrial, Mechatronic and Manufacturing Engineering*, v. 8, n. 3, p. 601–605.
- Pritsker, A Alan B; Waiters, Lawrence J e Wolfe, Philip M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management science*, v. 16, n. 1, p. 93–108.
- Roy, Ranjit K. (2010). *A primer on the Taguchi method*. Society of Manufacturing Engineers, Segunda edição.
- Shapiro, S.S. e Wilk, M. B. May(1965). An analysis of variance test for normality. *Biometrika*, v. 52, n. 3-4, p. 591.
- Słowiński, Roman; Soniewicki, Bolesław e Weglarz, Jan. (1994). DSS for multiobjective project scheduling. *European Journal of Operational Research*, v. 79, n. 2, p. 220–229.
- Soliman, Omar S e Elgendi, Elshimaa. (2014)a. A hybrid estimation of distribution algorithm with random walk local search for multi-mode resource-constrained project scheduling problems. *International Journal of Computer Trends and Technology (IJCTT)*, v. 8, n. 2, p. 57–64.
- Soliman, Omar S e Elgendi, Elshimaa. (2014)b. Niching estimation of distribution algorithm based on fuzzy clustering for multi-mode resource-constrained project scheduling problems. *International Journal of Scientific & Engineering Research*, v. 5, n. 3, p. 183–189.
- Sonmez, Rifat e Uysal, Furkan. (2014). Backward-forward hybrid genetic algorithm for resource-constrained multiproject scheduling problem. *Journal of Computing in Civil Engineering*, v. 29, n. 5, p. 1–9.
- Sprecher, Arno. (1994). Resource-constrained project scheduling: Exact methods for the multi-mode case. *Lecture Notes in Economics and Mathematics*, v. 409, p. 142.
- Sprecher, Arno e Drexl, Andreas. (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, v. 107, n. 2, p. 431–450.
- Sprecher, Arno; Hartmann, Sönke e Drexl, Andreas. (1997). An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum*, v. 19, n. 3, p. 195–203.
- Sprecher, Arno; Kolisch, Rainer e Drexl, Andreas. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 80, n. 1, p. 94–102.
- Sun, Kai e Yang, Gen-ke. (2008). Hybrid artificial immune system and extremal optimization algorithm for permutation flowshop scheduling problem. *Journal of Shanghai University (English Edition)*, v. 12, p. 352–357.

- Talbot, F Brian. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, v. 28, n. 10, p. 1197–1210.
- Tchao, Celso e Martins, Simone L. (2008). Hybrid heuristics for multi-mode resource-constrained project scheduling. Maniezzo, Vittorio; Battiti, Roberto e Watson, Jean-Paul, editors, *In Learning and Intelligent Optimization*, volume 5313, p. 234–242. Springer.
- Toffolo, Túlio AM; Santos, Haroldo G; Carvalho, Marco AM e Soares, Janniele A. (2013). An integer programming approach to the multimode resource-constrained multiproject scheduling problem. *Journal of Scheduling*, p. 1–13.
- Truong, Vu Xuan e Kachitvichyanukul, Voratas. (2007). A hybrid PSO algorithm for multi-mode resource-constrained project scheduling problems. *Proceedings of the 8th Asia Pacific Industrial Engineering and Management System Conference (APIEMS 2007)*, p. 1, (2007).
- Tseng, Lin-Yu e Chen, Shih-Chieh. (2009). Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem. *Evolutionary Computation on IEEE Transactions*, v. 13, n. 4, p. 848–857.
- Valls, Vicente; Ballestín, Francisco e Quintanilla, Sacramento. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, v. 165, n. 2, p. 375–386.
- Van Peteghem, Vincent e Vanhoucke, Mario. (2008). A genetic algorithm for the multi-mode resource-constrained project scheduling problem. v. 494, n. 8, p. 21.
- Van Peteghem, Vincent e Vanhoucke, Mario. (2009). An artificial immune system for the multi-mode resource-constrained project scheduling problem. Cotta, Carlos e Cowling, Peter, editors, *In Evolutionary computation in Combinatorial Optimization*, volume 5482, p. 85–96. Springer.
- Van Peteghem, Vincent e Vanhoucke, Mario. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 201, n. 2, p. 409–418.
- Van Peteghem, Vincent e Vanhoucke, Mario. (2011). Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *Journal of Heuristics*, v. 17, n. 6, p. 705–728.
- Van Peteghem, Vincent e Vanhoucke, Mario. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research*, v. 235, n. 1, p. 62–72.
- Wang, Ling e Fang, Chen. (2011). An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. *Information Sciences*, v. 181, n. 20, p. 4804–4822.

- Wang, Ling e Fang, Chen. (2012). An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research*, v. 39, n. 2, p. 449–460.
- Wauters, Tony; Kinable, Joris; Smet, Pieter; Vancroonenburg, Wim; Vanden Berghe, Greet e Verstichel, Jannes. (2014). The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling*, v. 19, n. 3, p. 271–283.
- Wauters, Tony; Verbeeck, Katja; Berghe, G Vanden e De Causmaecker, Patrick. (2011). Learning agents for the multi-mode project scheduling problem. *Journal of the Operational Research Society*, v. 62, n. 2, p. 281–290.
- Węglarz, Jan; Józefowska, Joanna; Mika, Marek e Waligóra, Grzegorz. (2011). Project scheduling with finite or infinite number of activity processing modes - A survey. *European Journal of Operational Research*, v. 208, n. 3, p. 177–205.
- Wu, Changzhi; Wang, Xiangyu e Lin, Jiang. (2014). Optimizations in project scheduling: A state-of-art survey. *Optimization and Control Methods in Industrial Engineering and Construction*, p. 161–177. Springer.
- Zhang, Hong; Tam, CM e Li, Heng. (2006). Multimode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering*, v. 21, n. 2, p. 93–103.
- Zhang, Lieping; Luo, Yingxiong e Zhang, Yu. (2015). Hybrid particle swarm and differential evolution algorithm for solving multimode resource-constrained project scheduling problem. *Journal of Control Science and Engineering*, v. 2015, p. 48.
- Zhu, Guidong; Bard, Jonathan F e Yu, Gang. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing*, v. 18, n. 3, p. 377–390.

Apêndice A

Tabela com respostas obtidas pelas combinações de parâmetros do algoritmo CLONALG-MRCPSP

Tabela A.1: Média do valores de GAP para cada combinação de parâmetro do algoritmo CLONALG

Experimento	POP	$N1$	$N2$	β	GAP_{avr}^1	GAP_{avr}^2
#1	300	70	15	10	2.5822	2.5639
#2	300	50	15	10	1.2431	1.2832
#3	300	70	10	10	2.5611	2.5254
#4	300	50	20	10	1.3182	1.3053
#5	200	50	15	3	1.0011	1.0006
#6	300	70	10	5	1.1378	1.1286
#7	200	50	20	5	0.9838	0.9985
#8	100	100	10	5	0.9020	0.8962
#9	300	50	20	3	1.2345	1.2443
#10	300	100	10	10	3.3284	3.4209
#11	300	50	10	5	1.0566	1.0409
#12	100	100	20	5	0.9475	0.9166
#13	300	50	15	3	1.1488	1.1460
#14	300	70	10	3	1.1209	1.1315
#15	200	100	10	10	0.9535	0.9298
#16	100	50	15	10	0.8404	0.8803
#17	100	100	10	10	0.8956	0.8671
#18	200	50	15	10	0.9428	0.9737
#19	100	70	15	5	0.8687	0.8731
#20	300	50	10	10	1.2354	1.2542
#21	200	70	10	10	1.0212	1.0620
#22	100	50	15	3	0.9621	0.9573
#23	300	50	15	5	1.0657	1.1009
#24	100	50	10	3	0.9423	0.9175
#25	200	100	15	5	0.8806	0.8852
#26	300	50	20	5	1.1452	1.1548

#27	100	70	15	10	0.8919	0.8693
#28	200	70	15	5	0.9963	0.9710
#29	100	70	15	3	0.9122	0.9280
#30	200	50	10	5	0.8868	0.9109
#31	100	100	20	10	0.9042	0.8725
#32	200	70	20	10	1.0980	1.0901
#33	100	100	15	3	0.9243	0.9425
#34	300	70	15	3	1.2012	1.2293
#35	200	70	15	3	1.0186	1.0298
#36	200	50	10	3	0.9540	0.9661
#37	300	100	10	3	1.0172	1.0285
#38	200	50	20	3	1.0707	1.0691
#39	300	100	15	3	1.0737	1.0639
#40	300	70	20	5	1.2365	1.2436
#41	300	70	20	10	2.5806	2.6529
#42	100	70	20	10	0.8918	0.8753
#43	100	70	20	3	0.9520	0.9752
#44	100	100	15	10	0.8849	0.8963
#45	200	100	20	5	0.9413	0.9139
#46	300	100	20	10	3.3730	3.4297
#47	100	70	10	10	0.8678	0.8810
#48	300	100	10	5	1.0408	1.0309
#49	200	100	10	3	0.8545	0.8715
#50	200	70	20	5	1.0419	1.0259
#51	200	70	15	10	1.0738	1.0635
#52	200	50	10	10	0.9516	0.9152
#53	200	70	10	5	0.9266	0.9423
#54	300	100	15	5	1.0780	1.0887
#55	300	100	15	10	3.4112	3.4078
#56	300	50	10	3	1.0565	1.0625
#57	200	70	10	3	0.9523	0.9658
#58	100	100	20	3	0.9872	0.9535
#59	100	100	10	3	0.9514	0.9614
#60	300	100	20	5	1.1526	1.1304
#61	100	70	20	5	0.9099	0.8951
#62	100	100	15	5	0.9232	0.9199
#63	100	50	20	3	0.9976	1.0117
#64	100	50	20	10	0.8734	0.8717
#65	200	100	20	10	0.9714	0.9542
#66	200	100	15	10	0.9557	0.9574
#67	100	50	10	5	0.8735	0.8718
#68	200	50	20	10	0.9899	0.9752
#69	100	50	15	5	0.8842	0.9223
#70	200	100	20	3	0.9841	0.9710
#71	100	70	10	3	0.9114	0.8773
#72	300	70	20	3	1.2802	1.2942
#73	300	70	15	5	1.1767	1.1977

#74	200	50	15	5	0.9283	0.9235
#75	100	50	10	10	0.8752	0.8616
#76	100	70	10	5	0.8389	0.8485
#77	100	50	20	5	0.9140	0.9334
#78	200	70	20	3	1.0679	1.0742
#79	200	100	10	5	0.8651	0.8805
#80	300	100	20	3	1.1639	1.1608
#81	200	100	15	3	0.9309	0.9168

Apêndice B

Boxplots para combinações de ordem 2 e 3 entre os parâmetros do algoritmo CLONALG-MRCPSP

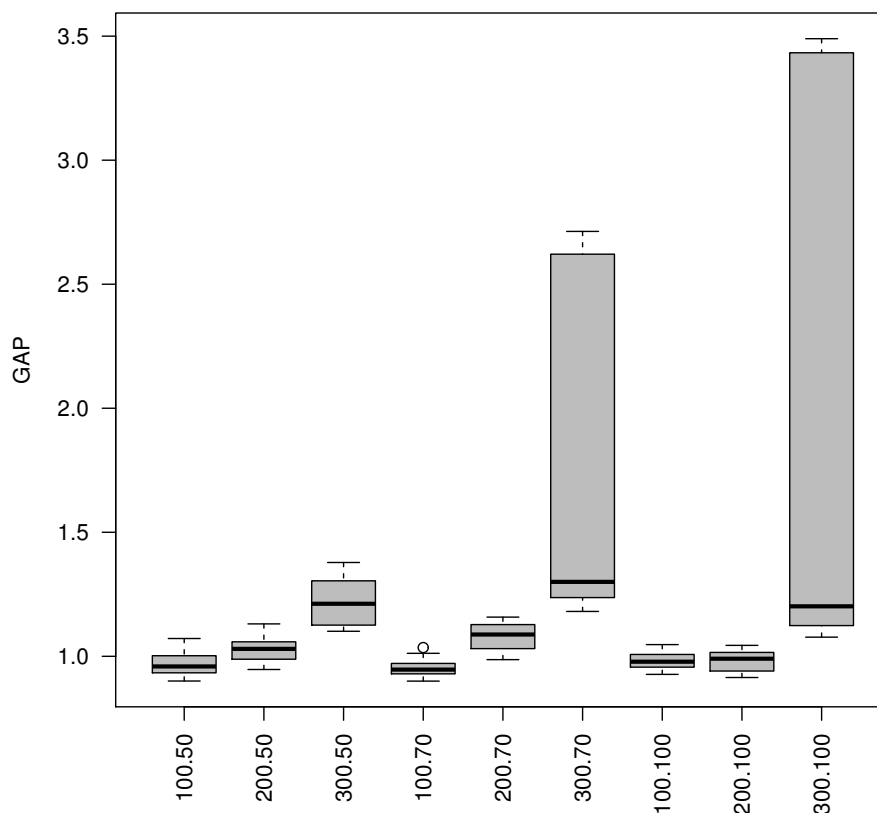


Figura B.1: Boxplot para comparações entre as combinações ($POP : N1$)

Percebe-se destaque para as combinações que possuem $POP = 100$ como aquelas que apresentam menor variância e mediana. Quando $POP = 300$, nota-se maior

variância e medianas. Já o parâmetro $N1$ possui pouca influência, pois apresenta soluções com pouca variância e baixa mediana para as três variações de níveis.

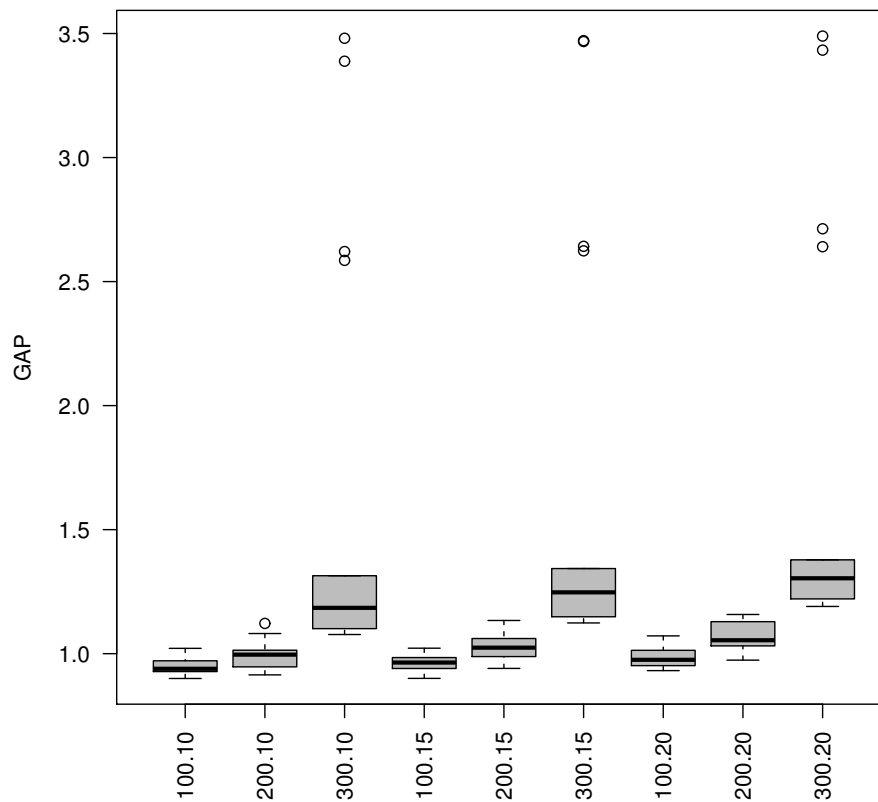


Figura B.2: Boxplot para comparações entre as combinações ($POP : N2$)

Claramente destaca-se o nível $POP = 100$, com menores medianas e variâncias. Os três melhores boxplots pertencem as combinações (100,10), (100,15) e (100,20). Logo, $N2$ possui pouca influência nessa combinação. Para reforçar essa análise, percebe-se que ao variar POP e mantendo $N1$ fixo, os valores de GAP crescem mais rápido do que variar $N1$ e manter POP fixo.

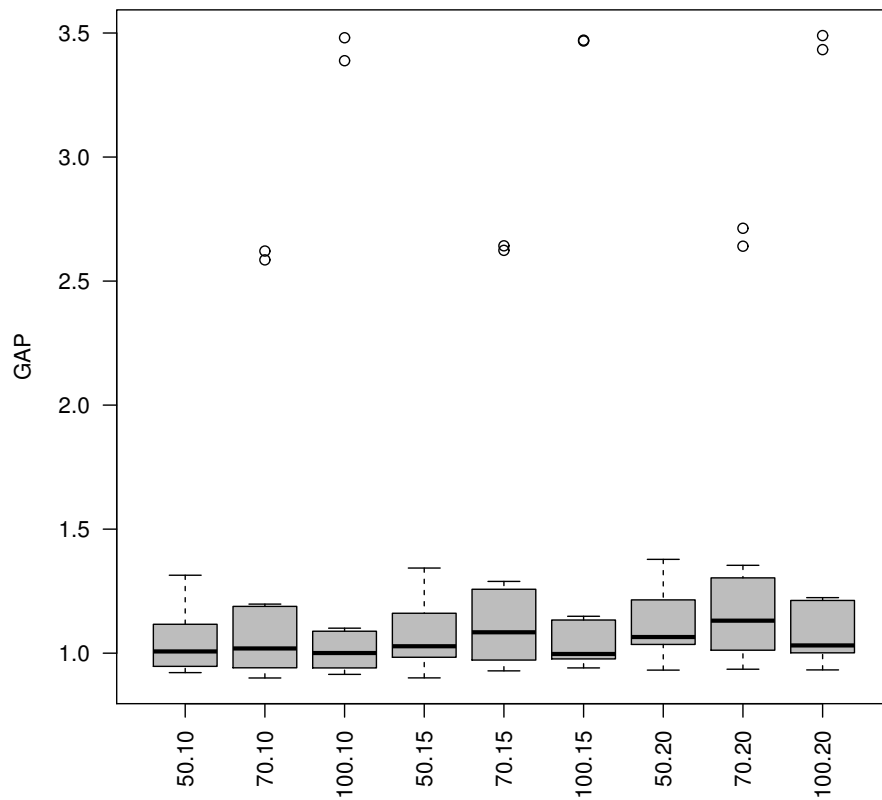


Figura B.3: Boxplot para comparações entre as combinações ($N1 : N2$)

Nessa comparação, os boxplots apresentam mediana semelhante com pouca diferença na variância entre os valores de GAP . Isso reforça a análise de que a combinação de ambos os parâmetros possui fraca influência (baixa interação).

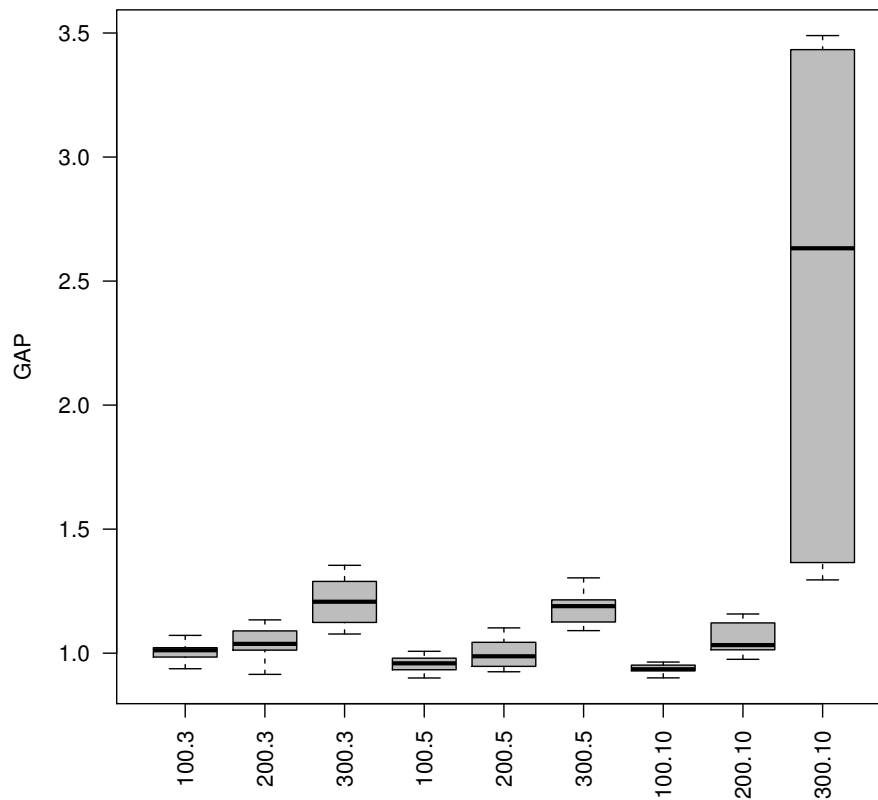


Figura B.4: Boxplot para comparações entre as combinações ($POP : \beta$)

Claramente a combinação 300:10 apresenta a maior mediana entre os valores de GAP , assim como a maior variância. Nota-se que ao variar POP e manter β fixo, há um maior crescimento dos valores de GAP . Quando $POP = 300$, independente do valor de β , foram obtidos valores de GAP maiores do que as outras combinações. Porém, para $POP = 100$ foram apresentados os menores valores de GAP .

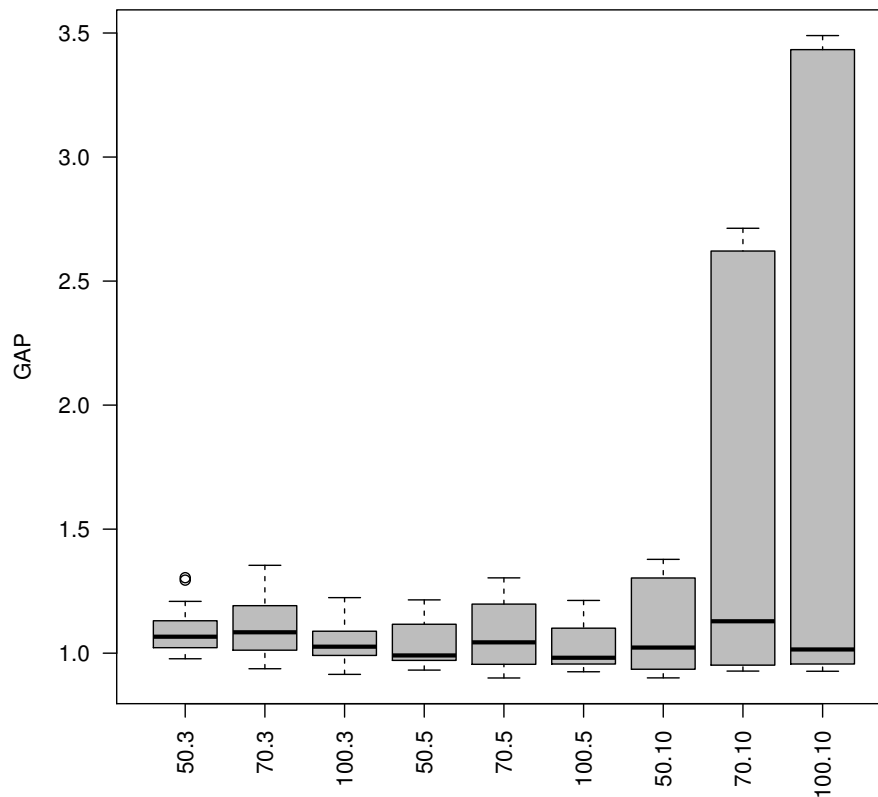


Figura B.5: Boxplot para comparações entre as combinações ($N1 : \beta$)

Percebe-se que em todas as combinações de níveis, são obtidos valores de GAP , para o 1º quartil dos boxplots, semelhantes, assim como medianas. As maiores variâncias acontecem quando $\beta = 10$, porém os outros dois níveis apresentam variâncias e medianas semelhantes. Pelo boxplot não é possível destacar qual nível de $N1$ apresenta os menores valores de GAP .

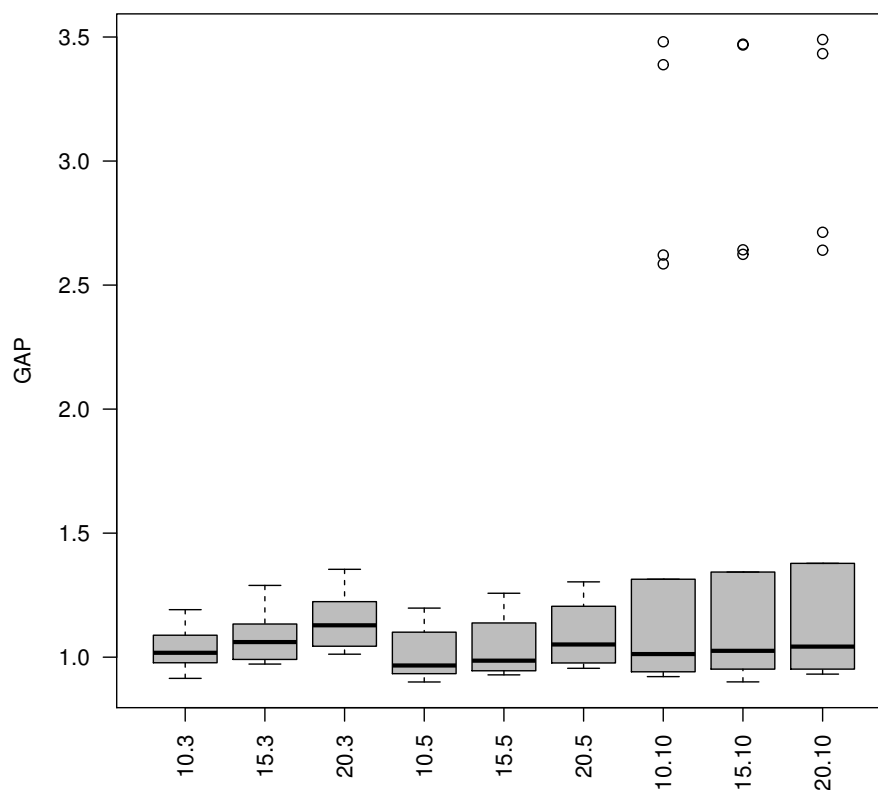


Figura B.6: Boxplot para comparações entre as combinações ($N2 : \beta$)

Os maiores valores de GAP são obtidos quando $\beta = 10$ para qualquer nível de $N1$. Também, são apresentados *outliers* somente para $\beta = 10$. Entretanto, para qualquer combinação de níveis, as medianas são semelhantes e, pelo boxplot, não é possível inferir qual combinação apresenta os melhores valores de GAP .

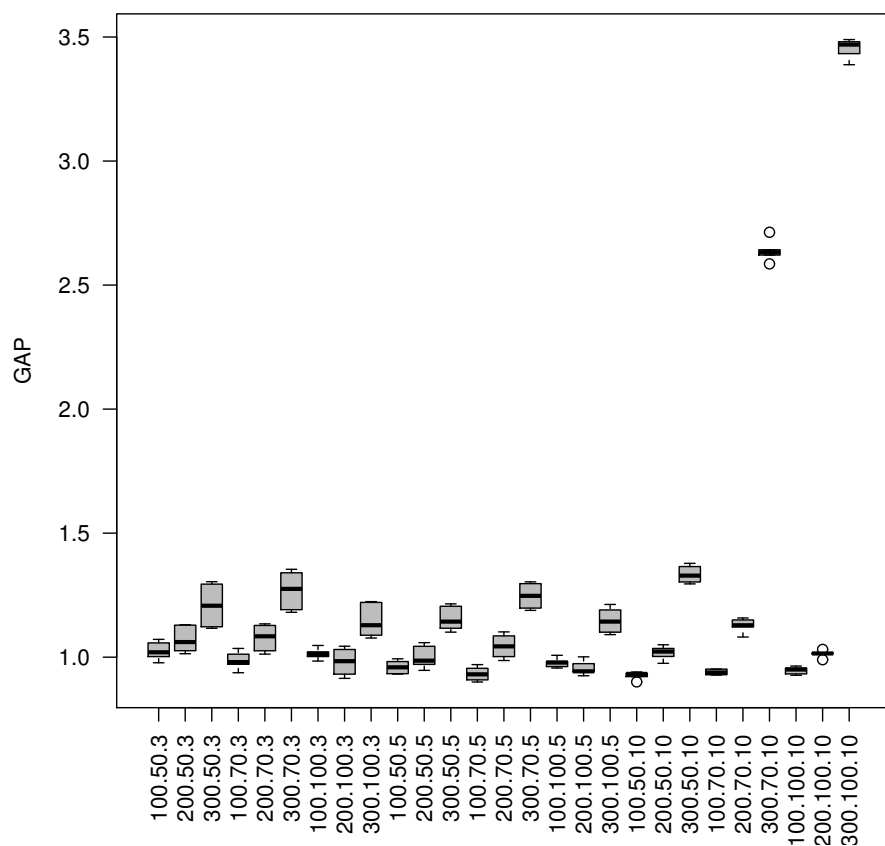


Figura B.7: Boxplot para comparações entre as combinações ($POP : N1 : \beta$)

Para essa combinação, novamente, destaca-se o nível $POP = 100$ como aquele que apresenta menores valores de GAP , assim como variâncias, independente dos níveis de $N1$ e β . As combinações com maior variância, acontecem quando $POP = 300$. Logo, a maior influência nessa combinação de três parâmetros, acontece pelo parâmetro POP .

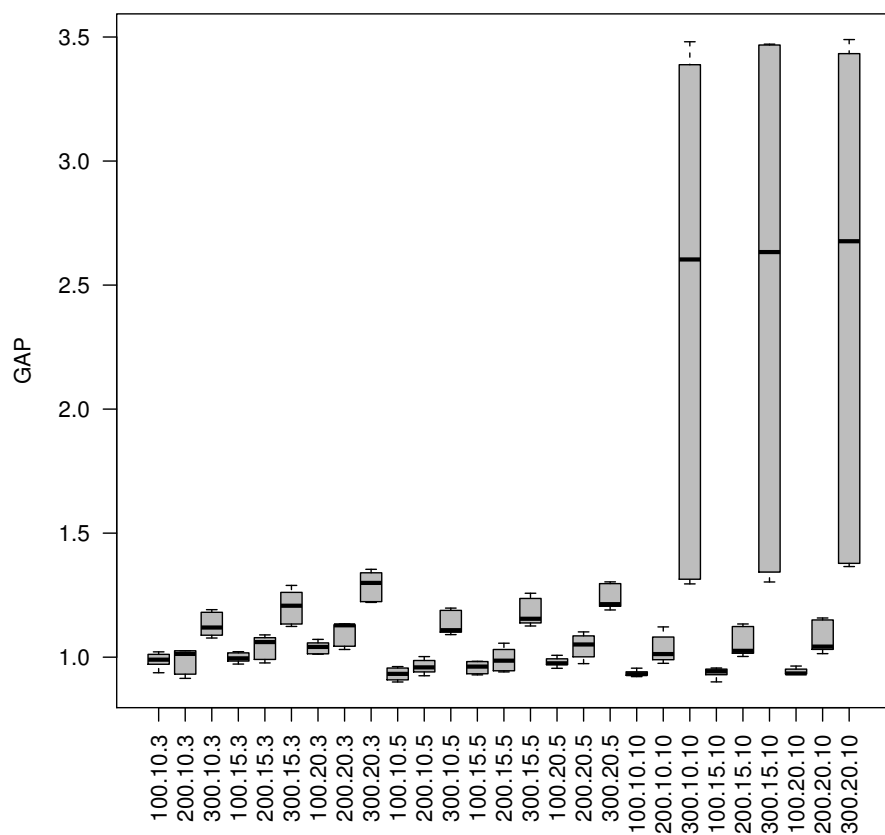


Figura B.8: Boxplot para comparações entre as combinações ($POP : N2 : \beta$)

As maiores variâncias e medianas são apresentadas quando $POP = 300$ e $\beta = 10$. Já as menores variâncias e medianas são obtidas quando $POP = 100$. Por essa combinação, nota-se que o nível $POP = 100$ e os níveis $\beta = 3$ ou $\beta = 5$ apresentam os melhores valores de GAP .

Apêndice C

Teste de Tukey para combinações de ordem 2 entre os parâmetros do algoritmo CLONALG-MRCPSP

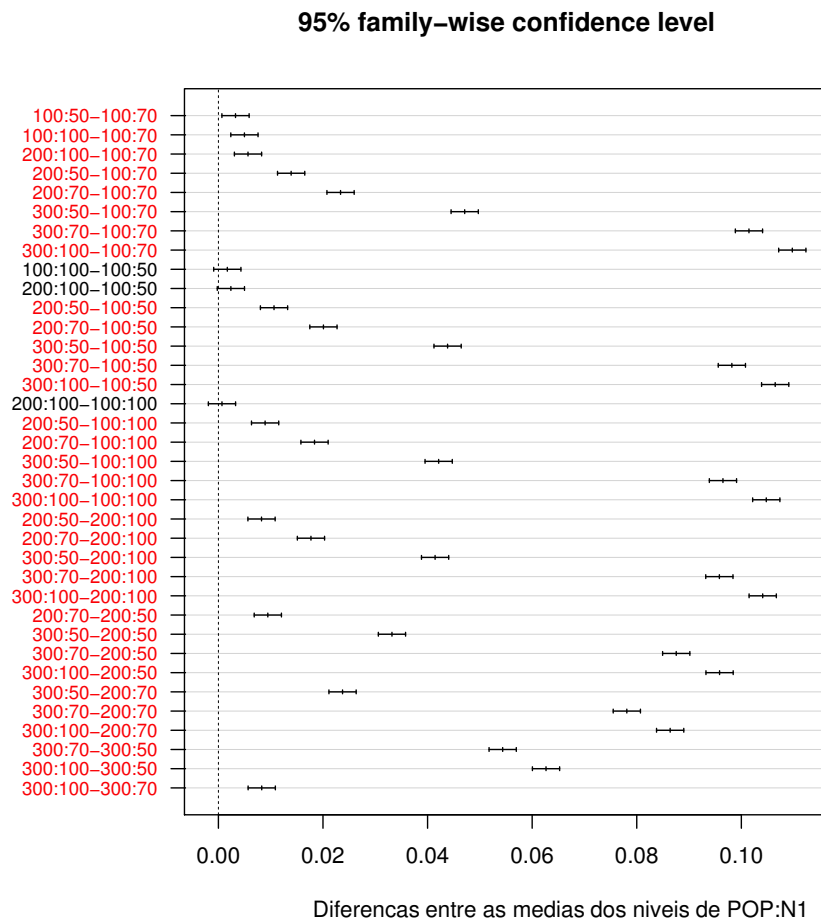


Figura C.1: Diferença para as médias das comparações entre os níveis dos parâmetros *POP* e *N1*. As comparações destacadas em vermelho apresentam diferenças estatísticas com 95% de confiança

As maiores diferenças são apresentadas para os intervalos mais afastados de zero. Nota-se que as maiores diferenças apresentam intervalos maiores que 0.08 e, que o nível $POP = 300$ aparece nessas diferenças. Já as menores diferenças apresentam o nível $POP = 100$ para os três níveis de $N1$, sendo que não houve diferença estatística entre as combinações $POP = 100 : N1 = 50$ e $POP = 100 : N1 = 100$. Isso indica que a combinação $POP = 100 : N1 = 70$ apresentou os menores valores de GAP , porém com pouca diferença para as combinações $POP = 100 : N1 = 50$ e $POP = 100 : N1 = 100$.

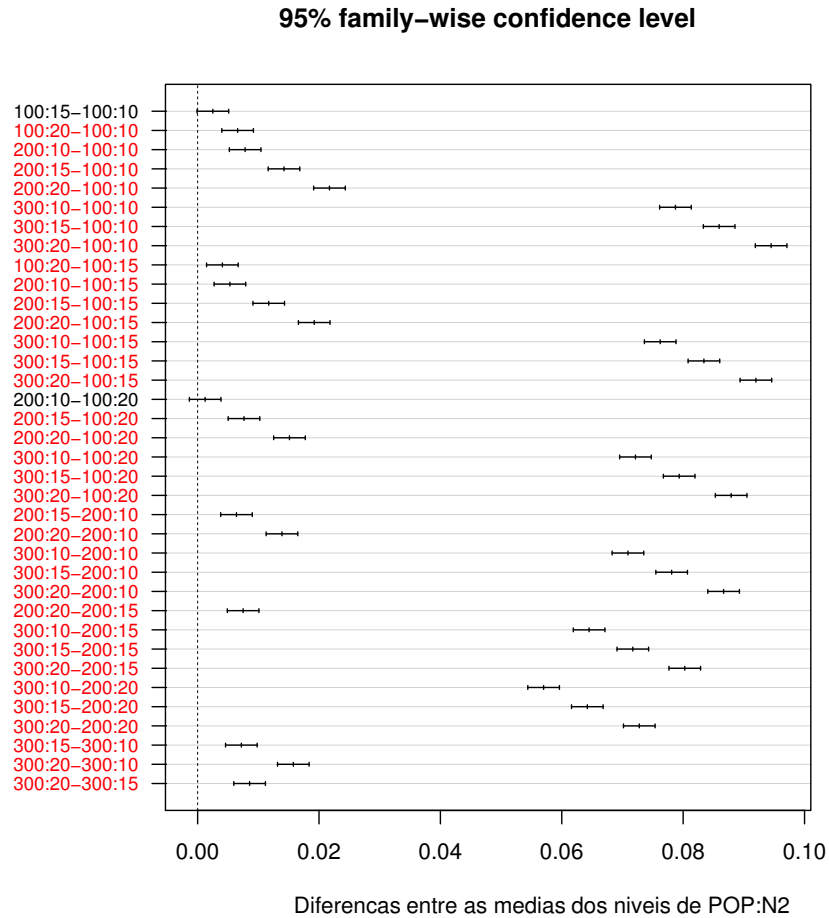


Figura C.2: Diferença para as médias das comparações entre os níveis dos parâmetros POP e $N2$

Não é apresentada diferença estatística para as combinações $POP = 100 : N2 = 10$ e $POP = 100 : N2 = 15$ e, também, essas duas combinações apresentam diferença estatística para todas as outras combinações possíveis. Logo, em complemento com a análise do boxplot da Figura B.2, $POP = 100 : N2 = 10$ e $POP = 100 : N2 = 15$ são as combinações que apresentaram melhores valores de GAP .

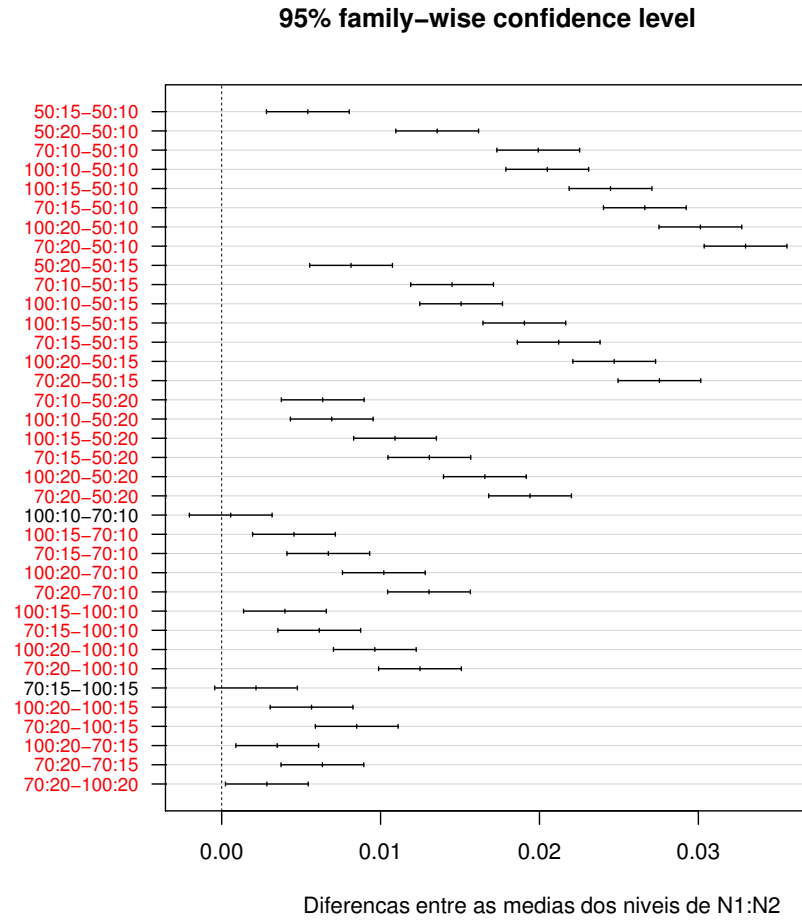


Figura C.3: Diferença para as médias das comparações entre os níveis dos parâmetros $N1$ e $N2$

O boxplot da Figura B.3, que compara os níveis desses parâmetros, apresentou semelhanças entre as medianas e valores de GAP obtidos. Esse resultado é percebido pelas diferenças estarem próximas de zero, sendo a maior diferença com valor próximo de 0.03. Essa análise também é percebida pelo gráfico de interação entre esses parâmetros. Portanto, ainda que o teste de Tukey tenha apontado diferenças estatísticas entre as combinações de $N1$ e $N2$, os valores de GAP obtidos por essas combinações são próximos, com leve destaque para as combinações $N1 = 50 : N2 = 10$, $N1 = 100 : N2 = 10$ e $N1 = 100 : N2 = 15$.

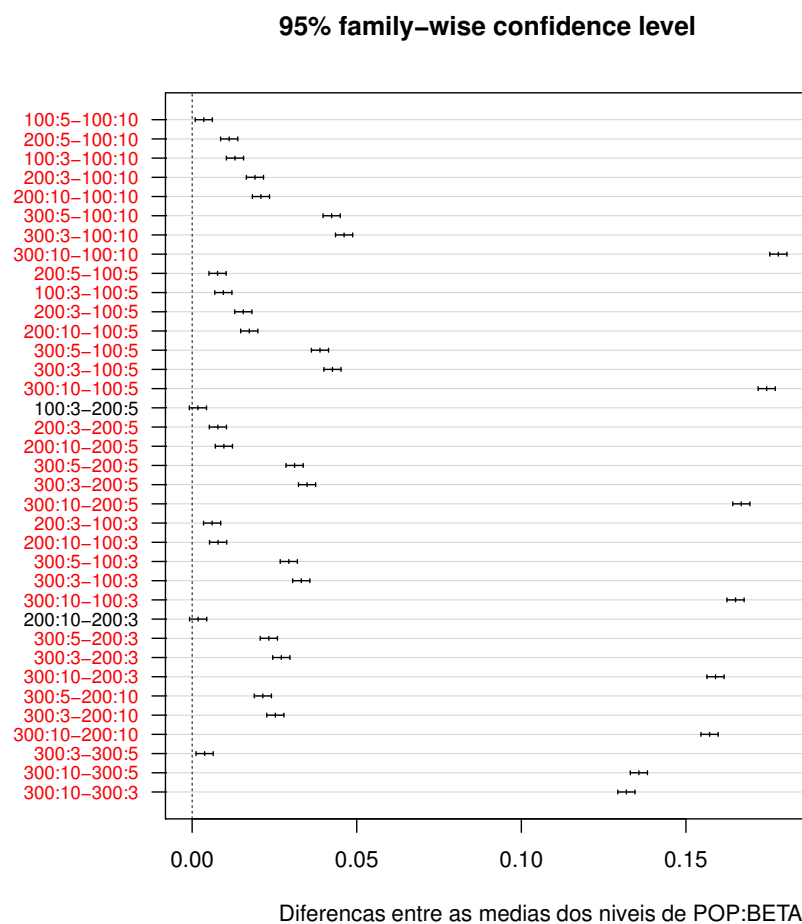


Figura C.4: Diferença para as médias das comparações entre os níveis dos parâmetros POP e β

Como apontado pelo boxplot da Figura B.4, quando $POP = 300$ foram obtidos os maiores valores de GAP . Essa análise é confirmada ao observar que as maiores diferenças acontecem quando $POP = 300$. Pelo teste de Tukey, foi confirmado que $POP = 100$ fornece as melhores respostas, sendo a melhor combinação $POP = 100 : \beta = 10$.

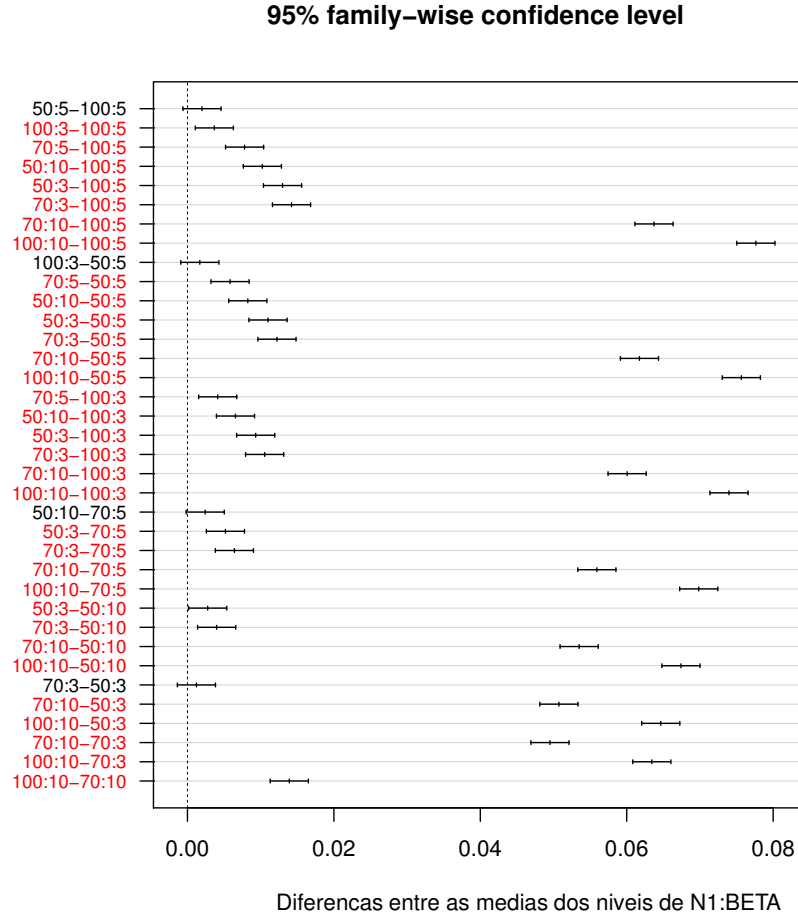


Figura C.5: Diferença para as médias das comparações entre os níveis dos parâmetros $N1$ e β

De acordo com as maiores diferenças em complemento com o boxplot da Figura B.5, quando $\beta = 10$, para essa combinação, são obtidos os maiores valores de GAP . As menores medianas foram obtidas para $N1 = 50 : \beta = 5$ e $N1 = 100 : \beta = 5$. Como é possível verificar, não houve diferença estatística entre as médias dessas comparações. A menor variância foi obtida para $N1 = 100 : \beta = 3$ e, também, verifica-se que não há diferença entre $N1 = 100 : \beta = 3$ e $N1 = 50 : \beta = 5$, que existe diferença entre $N1 = 100 : \beta = 3$ e $N1 = 100 : \beta = 5$, porém da ordem de 0.01. Então, destacam-se essas três combinações como aquelas que fornecem menores valores de GAP .

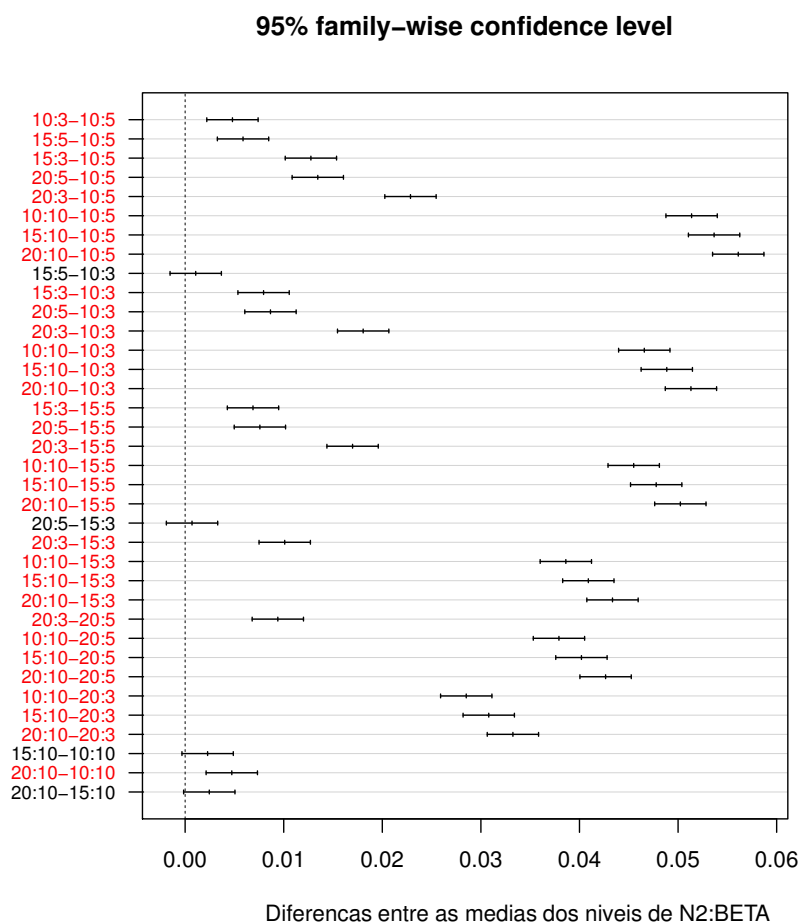


Figura C.6: Diferença para as médias das comparações entre os níveis dos parâmetros $N2$ e β

Pelo boxplot da Figura B.6, as melhores combinações foram $N2 = 10 : \beta = 3$ e $N2 = 10 : \beta = 5$. O teste de Tukey apresenta pequena diferença entre as médias dessas combinações. Portanto, as combinações $N2 = 10 : \beta = 3$ e $N2 = 10 : \beta = 5$ apresentam os menores valores de GAP , com pequeno destaque para $N2 = 10 : \beta = 5$.