



CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em
Modelagem Matemática e Computacional

VARIANTES DA MATHEURÍSTICA
LOCAL BRANCHING PARA O
PROBLEMA DE SEQUENCIAMENTO EM
PROJETOS COM RESTRIÇÃO EM
RECURSOS E MÚLTIPLOS
MODOS DE EXECUÇÃO

Belo Horizonte
Março de 2022



CENTRO FEDERAL DE EDUCAÇÃO
TECNOLÓGICA DE MINAS GERAIS

Diretoria de Pesquisa e Pós-Graduação

Programa de Pós-Graduação em
Modelagem Matemática e Computacional

VARIANTES DA MATHEURÍSTICA LOCAL BRANCHING PARA O PROBLEMA DE SEQUENCIAMENTO EM PROJETOS COM RESTRIÇÃO EM RECURSOS E MÚLTIPLOS MODOS DE EXECUÇÃO

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

Aluno : Gustavo Alves Fernandes

Orientador : Prof. Dr. Sérgio Ricardo de Souza

Belo Horizonte
Março de 2022

Fernandes, Gustavo Alves
F363v Variantes da metaheurística local branching para o problema de
sequenciamento em projetos com restrição em recursos e múltiplos modos de
execução / Gustavo Alves Fernandes. – 2022.
77 f.

Tese de doutorado apresentada ao Programa de Pós-Graduação em
Modelagem Matemática e Computacional.

Orientador: Sérgio Ricardo de Souza.

Tese (doutorado) – Centro Federal de Educação Tecnológica de Minas
Gerais.

1. Programação de produção – Teses. 2. Programação heurística – Teses.
3. Processos de ramificação – Teses. 4. Agenda de execução (Administração) –
Teses. 5. Otimização combinatória – Teses. I. Souza, Sérgio Ricardo de.
II. Centro Federal de Educação Tecnológica de Minas Gerais. III. Título.

CDD 005.1



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

**“VARIANTES DA MATHEURÍSTICA LOCAL BRANCHING PARA O
PROBLEMA DE SEQUENCIAMENTO EM PROJETOS COM
RESTRIÇÃO EM RECURSOS E MÚLTIPLOS MODOS DE EXECUÇÃO”.**

Tese de Doutorado apresentada por **Gustavo Alves Fernandes**, em 30 de março de 2022, ao Programa de Pós-Graduação em Modelagem Matemática e Computacional do CEFET-MG, e aprovada pela banca examinadora constituída pelos professores:

Prof. Dr. Sérgio Ricardo de Souza

Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Alexandre Xavier Martins

Universidade Federal de Ouro Preto

Prof. Dr. Haroldo Gambini Santos

Amazon.com

Prof. Dr. Mauricio Cardoso de Souza

Universidade Federal de Minas Gerais

Profª. Drª. Elisângela Martins de Sá

Centro Federal de Educação Tecnológica de Minas Gerais

Prof. Dr. Gustavo Campos Menezes

Centro Federal de Educação Tecnológica de Minas Gerais

Visto e permitida a impressão,

Profª. Drª. Elizabeth Fialho Wanner

Presidenta do Colegiado do Programa de Pós-Graduação em
Modelagem Matemática e Computacional

AGRADECIMENTOS

O autor gostaria de agradecer:

- aos pais, pela contínua preocupação representada pela frase “já terminou o doutorado?”;
- à esposa, pela paciência e carinho durante muitos momentos de desespero;
- à filha, que, por meio de um sorriso, injeta ânimo;
- aos professores do MMC, em especial ao professor Sérgio Ricardo de Souza;
- aos colegas do MMC, pelos esclarecimentos de “pequenas grandes” dúvidas;
- à comunidade *Stack Overflow*, pelos esclarecimentos indiretos de algumas “picuinhas”;
- às agências CAPES, FAPEMIG e CNPq, pelo apoio financeiro ao longo do mestrado e em um período de doutoramento.

Resumo

O Problema de 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. Trata-se de um problema NP-Difícil. Esta tese analisa sua resolução por uma estratégia de geração de pseudo-cortes, hibridizada com uma heurística de busca local, o que caracteriza em um método chamado de matheurística. A proposta apresenta duas novas definições para o pseudo-corte linear conhecido como *local branching cut*, bem como verifica outras duas definições da literatura para a resolução do MRCPSP por meio desse pseudo-corte. Também são analisados a influência dos parâmetros dessa matheurística e o impacto da heurística de busca local. Foram realizados testes computacionais utilizando instâncias das bibliotecas PSPLIB e MMLIB. Os melhores resultados conhecidos para as instâncias usadas nos testes são apresentados como referência para validar os resultados obtidos pela abordagem proposta nesta tese.

PALAVRAS-CHAVE: Sequenciamento em Projetos. *Matheurística*. *Local Branching*. Heurísticas. Problemas de Sequenciamento. Otimização Combinatória.

Abstract

The Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) aims to schedule all project jobs, subject to the precedence of these jobs, which, once started, can not be interrupted. The jobs have several execution modes combinations, consisting of different arrangements of processing time and resources consumption. The goal is to minimize the total project duration, finding start times for each project job. MRCPSP is an NP-hard problem. This thesis proposed a Local Branching variant method for solving it, which consists of a cut generation scheme hybridized with local search procedures. Two new pseudo-cuts characterizations are introduced, and two well-known pseudo-cuts from the literature are also investigated. These pseudo-cuts are known as the local branching cuts. This thesis also evaluated the Local Branching parameters and their influence on the method behavior. Computational tests were performed using instances sets from the PSPLIB and MMLIB libraries. The state-of-the-art results from literature for these instances are used as references to validate the results obtained by the approaches proposed in this thesis.

KEYWORDS: Project Scheduling. Matheuristic. Local Branching. Heuristics. Scheduling problems. Combinatorial Optimization.

Sumário

Lista de Figuras	ix
Lista de Tabelas	xi
Lista de Algoritmos	xii
1 Introdução	1
1.1 Apresentação Geral	1
1.2 Objetivos: Geral e Específicos	2
1.3 Justificativa	3
1.4 Organização da Tese	3
2 Caracterização do MRCPSP	4
2.1 Introdução	4
2.2 Modelagem por Programação Inteira	5
2.2.1 Sequenciamento em Projetos com Restrição em Recursos	5
2.2.2 Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução	8
2.3 Modelagem por Heurísticas	9
2.3.1 Representação de um Projeto	9
2.3.2 Esquemas de Geração de Sequenciamento	12
2.3.3 Regras de Prioridade	18
3 Trabalhos Relacionados	24
3.1 Abordagens Exatas	24
3.2 Abordagens via Matheurísticas	26
3.3 Abordagens via (Meta)heurísticas	27
4 Referencial Teórico	32
4.1 <i>Branch-and-Bound</i> e <i>Branch-and-Cut</i>	32
4.2 Matheurísticas	34
4.2.1 A Estratégia <i>Local Branching</i>	34
5 Metodologia	40
5.1 Esquema de Matheurística Proposto	40
5.2 <i>Local Branching Cut</i>	41
5.2.1 Proposta de Caracterização dos Subproblemas	43
5.2.2 <i>Strong Diversification</i>	46
5.3 Construção de uma Solução Inicial	47
5.4 Heurísticas de Busca Local	48

5.4.1	Estruturas de Vizinhaça	48
6	Resultados Computacionais	52
6.1	Resultados para o LB-B	53
6.1.1	<i>Tuning</i> dos parâmetros	53
6.1.2	Caracterizações do pseudo-corte	54
6.2	Resultados para o LB-I	57
6.2.1	<i>Tuning</i> dos parâmetros	57
6.2.2	Resultados em relação à Literatura	58
7	Considerações Finais	64
7.1	Propostas para Trabalhos Futuros	65
7.2	Publicações	65
	Referências	67

Lista de Figuras

2.1	Exemplo de projeto em estrutura de rede AoN.	10
2.2	Exemplo de codificação em listas (parcial) para o problema de projeto representado pela Figura 2.1.	12
2.3	Exemplo de sequenciamento obtido pelo SSS a partir da representação dada pela Figura 2.2	14
2.4	Exemplo de codificação em listas para o problema de projeto representado pela Figura 2.1.	15
4.1	Uma árvore de subproblemas. O conjunto inicial F é particionado em F_1 e F_2 . F_1 não será explorado, pois apresenta um limite inferior maior do que a melhor solução conhecida. Em seguida, F_2 é particionado em F_3 e F_4	33
4.2	Esquema de enumeração da estratégia LB. Adaptado de Fischetti e Lodi (2003).	37
5.1	(a) Solução de referência $\bar{x}$ codificada em listas. (b) Solução $x^1 \in N^{16}(\bar{x})$ codificada em listas	43
5.2	(a) Solução de referência $\bar{x}$ codificada em listas. (b) Solução $x^2 \in N^{16}(\bar{x})$ codificada em listas	44
5.3	Solução corrente $L(x)$ - em (a) - levando a um vizinho $L(x^1)$ - em (b) - na vizinhança NJ-SH	50
5.4	Solução corrente $L(x)$ levando a um vizinho $L(x^1)$ na vizinhança NJM-SWCM	50
6.1	Análise no tempo de execução, bem como gap_{opt} , gap_{best} e número de nós, ao resolver os problemas PSPLIB J20 e J30, pelo método LB-B Expressão (5.6), para critério de parada com 3600 e 7200 segundos. . .	57

Lista de Tabelas

2.1	Notação para o problema MRCPPSP	6
2.2	Modos de execução para as atividades no exemplo de projeto apresentado pela Figura 2.1.	10
2.3	Exemplo de execução do Algoritmo 1 sobre o problema de projeto apresentado pela Figura 2.1	11
2.4	Exemplo de execução passo-a-passo do Algoritmo SSS.	14
2.5	Exemplo de execução passo-a-passo do Algoritmo PSS	17
2.6	Exemplos de regras de prioridade para modos de execução	18
2.7	Exemplos de Regras de prioridade baseadas em atividades	21
2.8	Exemplos de Regras de prioridade baseadas na representação de rede	21
2.9	Exemplos de Regras de prioridade baseadas no caminho crítico	21
2.10	Exemplos de Regras de prioridade baseadas em recursos	21
2.11	Exemplos de Regras de prioridade compostas	22
3.1	Heurísticas adotadas na resolução do MRCPPSP com minimização do makespan	31
6.1	PSPLIB classes	52
6.2	MMLIB classes	53
6.3	MMLIB+ classes	53
6.4	Primeira combinação de valores para os parâmetros.	54
6.5	Segunda combinação de valores para o parâmetro de interesse.	54
6.6	Resultados PSPLIB (J10-J30) para as Expressões (5.5 e 5.6).	55
6.7	Resultados PSPLIB (J10-J30) para as Expressões (5.7 e 5.8).	55
6.8	Resultados para PSPLIB (J20 e J30) pelas caracterizações das Expressões (5.6) e (5.7)	56
6.9	Análise no tempo de execução para PSPLIB (J20-J30) pela caracterização da Expressão (5.6).	56
6.10	Primeira combinação de valores para os parâmetros.	58
6.11	Segunda combinação de valores para os parâmetros.	58
6.12	Comparação entre os resultados para os problemas J20 e J30 da PSPLIB, por Zhu et al. (2006) via LB, G12Max e SCIPMax de Schnell e Hartl (2017), a LB-B e LB-I. O símbolo “—” se refere a valores não publicados.	59
6.13	Comparação entre os resultados para os problemas J50 e J100 da MMLIB, por Schnell e Hartl (2017), LB-B e LB-I.	60

6.14	Comparação entre os resultados para os problemas J50 e J100 da MMLIB, entre abordagens no estado-da-arte via metaheurísticas e a variante LB-I. Resultados da literatura referem-se a 5000 e 50000 sequenciamentos gerados. O símbolo “–” refere-se a valores não publicados.	61
6.15	Resultados para cada classe de problemas J50 MMLIB+. O formato apresentado da classe é X-Y-Z, sendo X a quantidade de modos de execução, Y a quantidade de recursos renováveis e Z a quantidade de recursos não-renováveis.	62
6.16	Resultados para cada classe de problemas J100 MMLIB+. O formato apresentado da classe é X-Y-Z, sendo X a quantidade de modos de execução, Y a quantidade de recursos renováveis e Z a quantidade de recursos não-renováveis.	62
6.17	Comparação entre os resultados para os problemas J50 e J100 da MMLIB+, entre abordagens no estado-da-arte via metaheurísticas e a variante LB-I. Resultados da literatura referem-se a 5000 e 50000 sequenciamentos gerados. O símbolo “–” refere-se a valores não publicados.	63

Lista de Algoritmos

1	Método do Caminho Crítico	11
2	Esquema de Geração de Sequenciamento Serial - (SSS)	14
3	Esquema de Geração de Sequenciamento Paralelo - (PSS)	16
4	Esquema de <i>Branch-and-Bound</i> – Problema de minimização	33
5	Local Branching - (LB)	38
6	Esquema de matheurística baseada em Local Branching	42
7	SimpleStrongDiversification	46
8	StrongDiversification	46
9	StrongDiversificationAtRandom	47
10	VND adaptado	51

Capítulo 1

Introdução

1.1 Apresentação Geral

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 *International Organization for Standardization* define, na norma ISO 8402, projeto como um processo único, 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, tese ou trabalho universitário; dentre outros.

[Demeulemeester \(2002\)](#) afirma que o 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 os 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 (tarefas) que devem ser executadas de acordo com um conjunto de restrições de precedência. 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 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 tese 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 uma abordagem baseada em matheurística para resolução do MRCPSP. A abordagem proposta possui um esquema de geração de pseudo-cortes. Serão apresentadas duas novas caracterizações de pseudo-cortes, que serão exploradas devido à influência desses pseudo-cortes durante o processo de enumeração. Os parâmetros da abordagem proposta serão analisados e discutidos, uma vez que têm grande influência na qualidade dos resultados obtidos.

1.2 Objetivos: Geral e Específicos

O objetivo geral desta tese é analisar e propor uma variação da matheurística *Local Branching* para o MRCPSP. Para atingir esse objetivo, os objetivos específicos a seguir devem ser satisfeitos:

- Apresentar e discutir a formulação matemática do MRCPSP;
- Apresentar a estratégia de geração de pseudo-cortes para a formulação MRCPSP;
- Realizar um estudo computacional sobre a influência desses pseudo-cortes, que são adicionados à formulação do MRCPSP;
- Realizar um estudo sobre a influência dos parâmetros da matheurística proposta;
- Resolver os problemas das classes de instâncias das bibliotecas públicas PSPLIB ([Kolisch e Sprecher, 1997](#)) e MMLIB ([Van Peteghem e Vanhoucke, 2014](#));

1.3 Justificativa

Este estudo justifica-se a partir do que é destacado por [Lodi \(2009\)](#), quando afirma que o desempenho de uma abordagem de geração de cortes depende da qualidade da desigualdade acrescentada, isto é, estratégias baseadas em *Branch-and-Cut* se diferem pelos cortes que são gerados ao longo da enumeração. Sendo assim, a análise da qualidade de cortes possui relevância para pesquisas em métodos baseados em geração de cortes. Mesmo que nesta tese sejam utilizados os pseudo-cortes, a motivação para o estudo destes é semelhante ao destacado em [Lodi \(2009\)](#) para cortes. Estende-se essa justificativa com a análise da influência de parâmetros, que também possui grande impacto no desempenho de procedimentos computacionais ([López-Ibáñez et al., 2016](#)). Por fim, a justificativa é complementada pela importância do problema para a indústria e literatura, conforme já discutido na apresentação geral desse capítulo introdutório, quando destaca-se a importância do contexto de projetos nas empresas, assim como será evidente o interesse no estudo desse problema pela literatura ao longo dos anos, destacado no Capítulo 3.

1.4 Organização da Tese

A tese está organizada da seguinte maneira. O Capítulo 2 apresenta a descrição do problema de interesse, bem como sua formulação recorrente na literatura, a notação utilizada nesta tese e alguns procedimentos específicos para sequenciamento em projetos. 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 dos procedimentos utilizados nesta tese para a solução do problema objeto de interesse. O Capítulo 5 apresenta as heurísticas, as caracterizações dos pseudo-cortes e a matheurística proposta nesta tese para resolução do MRCPSP. No Capítulo 6 são apresentados os resultados obtidos por configurações dos algoritmos, assim como comparações com os melhores trabalhos na literatura, sendo que instâncias públicas foram utilizadas para os testes computacionais. Finalmente, o Capítulo 7 apresenta as conclusões sobre o tema em discussão nesta tese.

Capítulo 2

Caracterização do MRCPSP

Este Capítulo apresenta a caracterização do Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (*Multi-mode Resource Constrained Project Scheduling Problem - MRCPSP*). Inicia-se, na Seção 2.1, com uma discussão sobre Otimização Combinatória e Inteira. Em seguida, na Seção 2.2, o MRCPSP será descrito por meio de sua estrutura matemática. A Seção 2.3 apresenta sua representação por heurísticas.

2.1 Introdução

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 ser uma tarefa, em vários casos, 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 *the greatest hits* 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.

Em Otimização Inteira, conforme exposto em [Bertsimas e Tsitsiklis \(1997\)](#), há o objetivo de se encontrar uma solução x^* sobre um conjunto discreto F que otimize uma função objetivo $c(x)$, para todo $x \in F$. Esse tipo de problema de otimização aparece em vários contextos da ciência e engenharia. Um problema de Otimização Linear Inteira é semelhante a um problema de Otimização Linear, exceto pela restrição de que todas as variáveis devem assumir somente valores inteiros. Em geral, seja a matriz A e os vetores b, c . Então, o problema:

$$\min \quad c'x \quad (2.1)$$

$$\text{s.a.} \quad Ax = b \quad (2.2)$$

$$x \geq 0 \quad (2.3)$$

$$x \text{ inteiro} \quad (2.4)$$

é conhecido como Problema de Programação Linear Inteira (PLI). Caso os valores assumidos pela variável x possam ser somente $x = 0$ ou $x = 1$, então o problema é conhecido como problema de Programação Linear Inteira Binária (PLIB). Problemas de programação inteira estão na classe de problemas NP-Difícil ([Jünger et al., 2009](#)).

2.2 Modelagem por Programação Inteira

Para a apresentação do problema de otimização discutido nesta tese, será adotada a notação apresentada pela Tabela 2.1. O problema será apresentado por meio de seu modelo matemático e, também, por meio de um grafo, como uma representação visual.

2.2.1 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 n tarefas, de modo que cada tarefa $j = 1, \dots, n$ deve ser processada sem interrupção, respeitando-se a relação de precedência entre elas, i.e., cada tarefa j pode ser iniciada somente após todas as suas predecessoras em $\mathcal{P}_j$ estarem finalizadas. As tarefas $j = 1$ e $j = n$ são fictícias, de modo que $j = 1$ deve ser a primeira tarefa a ser iniciada sem algum predecessor e $j = n$ deve ser a última tarefa com nenhum sucessor. Para cada tarefa j do projeto, existe um tempo de processamento d_j e uma demanda de recursos renováveis r_{jk}^ρ para essa tarefa. Durante a execução do projeto, para cada período de tempo t , existe uma quantidade fixa de recursos renováveis R_k^ρ . Deve-se ressaltar que as tarefas fictícias possuem tempo de duração e consumo de recursos iguais a zero. Todos os parâmetros são valores inteiros não negativos. Então, o objetivo do RCPSP é encontrar uma combinação de sequência entre as tarefas, resultando em um sequenciamento com tempos de início

Tabela 2.1: Notação para o problema MRCPS P

Símbolo	Definição
$j = 1, \dots, n$	tarefas de um projeto
$j = 1$	Primeira tarefa do projeto
$j = n$	Última tarefa do projeto
M_j	Conjunto de modos de execução da tarefa j
$m \in M_j$	Modo m de execução da tarefa j
$\mathcal{P}_j$	Conjunto de predecessores da tarefa j
$\mathcal{S}_j$	Conjunto de sucessores da tarefa j
d_{jm}	Tempo de duração de uma tarefa j no modo m
$k = 1, \dots, K^\rho$	Recursos do tipo renovável
$k = 1, \dots, K^\nu$	Recursos do tipo não-renovável
R_k^ρ	Quantidade disponível do recurso renovável k
R_k^ν	Quantidade disponível do recurso não renovável k
r_{jmk}^ρ	Quantidade do recurso renovável k consumida pela tarefa j no modo m
r_{jmk}^ν	Quantidade do recurso não renovável k consumida pela tarefa j no modo m
ST_j	Tempo de início de uma tarefa j
FT_j	Tempo de término de uma tarefa j
EST_j	Tempo de início mais cedo de uma tarefa j
EFT_j	Tempo de término mais cedo de uma tarefa j
LST_j	Tempo de início mais tardio de uma tarefa j
LFT_j	Tempo de término mais tardio de uma tarefa j
T	Horizonte do projeto
C_{max}	duração do projeto ou <i>makespan</i>

definidos, de modo a respeitar as relações de precedência e disponibilidade de recursos, tal que o *makespan* (duração do projeto – C_{max}) seja minimizado. A variável de decisão x_{jt} é igual a 1 se a tarefa 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:

$$\min \sum_{t=EFT_n}^{LFT_n} tx_{nt} \quad (2.5)$$

s.a.

$$\sum_{t=EFT_j}^{LFT_j} x_{jt} = 1, \quad j = 1, \dots, n \quad (2.6)$$

$$\sum_{t=EFT_j}^{LFT_j} (t - d_j)x_{jt} \geq \sum_{t=EFT_i}^{LFT_i} tx_{it}, \quad j = 1, \dots, n; i \in \mathcal{P}_j \quad (2.7)$$

$$\sum_{j=1}^n \sum_{s=t-d_j}^t r_{jk}^\rho x_{js} \leq R_k^\rho, \quad k = 1, \dots, K^\rho; t = 1, \dots, LFT_n \quad (2.8)$$

$$x_{jt} \in \{0, 1\}, \quad j = 1, \dots, n; t = EFT_j, \dots, LFT_j \quad (2.9)$$

Por essa formulação, a Expressão (2.5) apresenta o objetivo de minimizar o término da última tarefa, representada por x_{nt} . A Expressão (2.6) representa a atribuição de um único tempo de término por tarefa. A Expressão (2.7) apresenta a relação de precedência entre tarefas. A Expressão (2.8) assegura que a disponibilidade de cada recurso renovável por período não é violada. Finalmente, a Expressão (2.9) 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 , na Seção 2.3.1.1.

Embora o RCPSP possua um modelo aplicável em algumas situações, esse problema não cobre todas as que ocorrem na prática, conforme apontam [Hartmann e Briskorn \(2010\)](#). Consequentemente, diversas variações do RCPSP foram propostas, como:

- Sequenciamento preemptivo;
- Variação da requisição de recursos no tempo por cada tarefa;
- Tempo de preparo para cada tarefa ser iniciada (*setup time*);
- Múltiplos modos de execução para cada tarefa, sendo que cada modo consiste de tempo de duração e consumo de recursos;
- Datas de *release* para cada tarefa 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 tarefa por período;
- Capacidade de recursos variando com o tempo;
- Múltiplos objetivos, como, por exemplo, redução do tempo de atraso na entrega do projeto, redução da utilização dos recursos e redução do custo do sequenciamento.

No evento MISTA (*Multidisciplinary International Scheduling Conference - Theory & Applications*), em sua edição de 2013, uma nova variação do RCPSP foi proposta no *MISTA 2013 challenge*, que é um evento de competição interno ao MISTA, 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. \(2014\)](#), e descreve que cada projeto em separado possui um conjunto de tarefas e recursos, com o objetivo de minimizar o tempo total de duração ao sequenciar todas as tarefas de todos os projetos.

Revisões que apresentam, detalhadamente, variações relacionadas a tarefas, 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 tese é o Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução (MRCPSP).

2.2.2 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 deste último. A diferença está na existência de modos de execução nos quais cada tarefa $j = 1, \dots, n$ do projeto deve ser executada. Cada modo de execução $m \in M_j$ estabelece o tempo de processamento e a demanda de recursos para a tarefa. As tarefas fictícias possuem um único modo de execução, com tempo de duração e consumo de recursos nulos. O número de modos em que uma tarefa j pode ser executada é dado por $|M_j|$ e o tempo de processamento da tarefa 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 tarefa 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. Todos os parâmetros são valores inteiros não negativos. Portanto, o objetivo do MRCPSP é encontrar uma combinação dos modos de execução para todas as tarefas, assim como uma combinação da sequência entre essas tarefas, resultando em um sequenciamento com tempo de início e consumo de recursos para cada tarefa, de maneira que o *makespan* seja minimizado. A variável de decisão é denotada por x_{jmt} , sendo $x_{jmt} = 1$ se a tarefa j é executada no modo m e possui término no período t . Caso contrário, $x_{jmt} = 0$.

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

$$(P) \quad \min \quad \sum_{m \in M_n} \sum_{t=EFT_n}^{LFT_n} tx_{nmt} \quad (2.10)$$

s.a.

$$\sum_{m \in M_j} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1, j = 1, \dots, n \quad (2.11)$$

$$\sum_{m \in M_j} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm})x_{jmt} - \sum_{m \in M_i} \sum_{t=EFT_i}^{LFT_i} tx_{imt} \geq 0, j = 1, \dots, n; i \in \mathcal{P}_j \quad (2.12)$$

$$\sum_{j=1}^n \sum_{m \in M_j} \sum_{s=t-d_{jm}}^t r_{jmk}^\rho x_{jms} \leq R_{kt}^\rho, k = 1, \dots, K^\rho; t = 1, \dots, LFT_n \quad (2.13)$$

$$\sum_{j=1}^n \sum_{m \in M_j} \sum_{t=EFT_j}^{LFT_j} r_{jmk}^\nu x_{jmt} \leq R_k^\nu, k = 1, \dots, K^\nu \quad (2.14)$$

$$x_{jmt} \in \{0, 1\} (j = 1, \dots, n; m \in M_j; t = EFT_j, \dots, LFT_j) \quad (2.15)$$

Nessa formulação, a Expressão (2.10) apresenta o objetivo de minimizar o término da última tarefa, isto é, minimizar o *makespan*; já a Expressão (2.11) representa a execução de um modo por tarefa, assim como a atribuição de um único tempo de término; a Expressão (2.12) apresenta a relação de precedência entre tarefas; a Expressão (2.13) assegura que a disponibilidade de recursos renováveis por período não seja violada; a Expressão (2.14) corresponde à restrição de recursos não renováveis; finalmente, a Expressão (2.15) 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. Conforme já exposto, esses valores serão detalhados na Seção 2.3.1.1.

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 tarefas 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 tarefa) 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 tarefas com, no mínimo, três modos de execução, não podem ser solucionados na otimalidade por métodos exatos em tempo aceitável e sugerem que heurísticas ou métodos exatos hibridizados são boas abordagens para problemas com um maior número de modos de execução e maior número de tarefas.

2.3 Modelagem por Heurísticas

As heurísticas discutidas nesta seção são comuns na literatura para a descrição e a resolução do MRCPSP.

2.3.1 Representação de um Projeto

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 2.1 apresenta um exemplo dessa estrutura em rede, para um projeto com 8 tarefas, sendo 2 fictícias ($j = 1$ e $j = 8$). Já a Tabela 2.2 apresenta os modos de execução das atividades, com seus respectivos tempos de processamento e consumo de recursos, para $K = 2$ tipos de recursos, sendo $k = 1$ representando o recurso renovável e $k = 2$ representando o recurso não renovável.

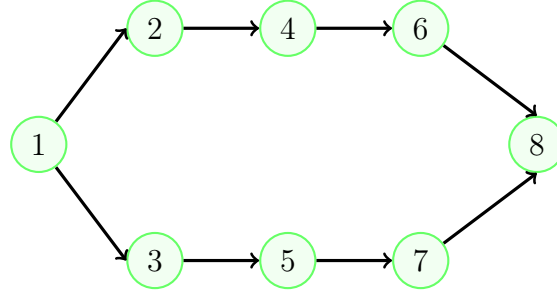


Figura 2.1: Exemplo de projeto em estrutura de rede AoN.

Tabela 2.2: Modos de execução para as atividades no exemplo de projeto apresentado pela Figura 2.1.

j	m	d_{jm}	r_{jmk}^ρ	r_{jmk}^ν	$R_k^\rho, k = 1$	$R_k^\nu, k = 2$
1	1	0	0	0	{4}	{15}
2	1	3	2	2		
	2	4	1	1		
3	1	2	3	4		
	2	4	2	1		
4	1	1	4	2		
	2	3	2	2		
5	1	2	2	3		
	2	2	3	2		
6	1	3	1	2		
	2	1	3	3		
7	1	4	3	1		
	2	6	1	1		
8	1	0	0	0		

2.3.1.1 Método do Caminho Crítico

O Método do Caminho Crítico (*Critical Path Method* - CPM), datado dos anos 1950 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 violar a 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.

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 restri-

Algoritmo 1: Método do Caminho Crítico

```

1 Inicializar  $EST_1 \leftarrow EFT_1 \leftarrow 0$  ;
2 para  $j \leftarrow 2$  até  $n$  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  $LFT_n \leftarrow LST_n \leftarrow EFT_n$  ;
7 para  $j \leftarrow n - 1$  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

```

ção 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 tarefa j , a folga de uma tarefa SLK_j é definida como:

$$SLK_j = LST_j - EST_j = LFT_j - EFT_j \quad (2.16)$$

As tarefas que possuem folga igual a zero são chamadas de tarefas críticas (Kelley Jr e Walker, 1959), o que significa que, se uma tarefa crítica não iniciar no tempo definido, irá atrasar o término do projeto. 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 viável com menor duração e executando o CPM, obtém-se o Caminho Crítico Minimal. Por outro lado, atribuindo a cada atividade o modo viável com maior duração, obtém-se o Caminho Crítico Maximal.

Como ilustração da execução do Algoritmo 1, a Tabela 2.3 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 , a partir dos dados apresentados na Tabela 2.2.

Tabela 2.3: Exemplo de execução do Algoritmo 1 sobre o problema de projeto apresentado pela Figura 2.1

j	m	d_{jm}	EST	EFT	LST	LFT	SLK
1	1	0	0	0	0	0	0
2	1	3	0	3	1	4	1
3	1	2	0	2	0	2	0
4	1	1	3	4	4	5	1
5	1	2	2	4	2	4	0
6	1	3	4	7	5	8	1
7	1	4	4	8	4	8	0
8	1	0	8	8	8	8	0

A Tabela 2.3 informa que o valor do caminho crítico é de 8 períodos de tempo para a atribuição de modos definida. Além do mais, as atividades pertencentes

ao caminho crítico são 1, 3, 5, 7 e 8, pois apresentam folga igual a zero. Outra informação útil presente nessa tabela é o valor final do *makespan*. Se todas as atividades críticas estiverem dispostas no tempo, dentro da janela $[EST, LFT]$, satisfazendo a disponibilidade de recursos renováveis ao longo dos 8 períodos de tempo, então o projeto, certamente, será finalizado com *makespan* igual a 8 unidades de tempo. Caso contrário, possivelmente, o projeto sofrerá atraso.

2.3.1.2 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).

Nesta tese, a codificação $S(x) = (L_J, L_M, L_T)$ representa uma solução x por meio de listas, ou seja, L_J para as tarefas, L_M para modos de execução e L_T para os tempos de término. Logo, essa codificação em listas indica que, durante o processamento de um sequenciador para uma tarefa j na posição i de L_J , seu modo de execução na posição i de L_M é verificado para atribuir um tempo de término na posição i de L_T .

A lista L_J deve ser viável em relação à restrição de precedência entre as tarefas. Logo, para uma tarefa j na posição i da lista de tarefas L_J , 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 2.2 apresenta um exemplo parcial desse esquema de codificação, sem a atribuição dos tempos de término, a partir do projeto apresentado pela Figura 2.1.

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1

Figura 2.2: Exemplo de codificação em listas (parcial) para o problema de projeto representado pela Figura 2.1.

Para construir a lista de tarefas apresentada pela Figura 2.2, a tarefa 1 é a primeira a entrar na lista. Então, as tarefas 2 e 3 estão elegíveis, pois todos os seus predecessores já foram selecionados para a lista L_J . Por algum mecanismo de escolha, determinou-se que a tarefa 2 será a próxima a entrar para a lista de tarefas. Em seguida, estão elegíveis as tarefas 3 e 4. O mesmo critério de escolha determinou que a tarefa 4 deverá ser a próxima para entrar na lista, fazendo com que as tarefas 3 e 6 sejam elegíveis. Seguindo este mecanismo, constrói-se a representação apresentada pela Figura 2.2. É importante destacar que as tarefas 1 e 8 devem ser, respectivamente, a primeira e a última tarefas da lista de tarefas. O mecanismo que determina a ordem de escolha entre as tarefas, bem como os modos de execução por tarefa, será abordado nas seções seguintes.

2.3.2 Esquemas de Geração de Sequenciamento

A partir de uma atribuição de modos de execução para cada tarefa 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 tarefas, é necessário sequenciar uma tarefa inicial e, em seguida, todas as suas sucessoras em algum instante após a tarefa inicial. Esse processo segue até que a última tarefa seja sequenciada.

Definição 2.3.1. *Um sequenciamento parcial é aquele em que um subconjunto I de tarefas já foram sequenciadas, para $I \subseteq J$.*

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. Para isso, um conjunto de tarefas elegíveis para o sequenciamento é construído, chamado de conjunto de decisão D_n . Quando uma tarefa é elegível, ou seja, está em D_n , significa que todos os seus predecessores já foram sequenciados. Esses métodos são descritos a seguir.

2.3.2.1 Método Serial

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

$$D_e = \{j : j \notin S_e, \mathcal{P}_j \subseteq S_e\}$$

A cada estágio e , uma tarefa $j \in D_e$ é selecionada por meio de algum critério de seleção, removida de D_e e, após, acrescentada em S_e , de modo que a tarefa j foi sequenciada. O método finaliza no estágio $e = n$, quando todas as tarefas do projeto estiverem em S_e .

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 (horizonte) para a duração do projeto e $v(j)$ um valor associado a $j \in D_e$. Então, Kolisch (1996b) apresenta o SSS de acordo com o Algoritmo 2.

Pela linha 4 no Algoritmo 2, uma tarefa j^* será escolhida como aquela que apresentar o menor valor de $v(j)$, $\forall j \in D_e$, 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 tarefas receberam um tempo de término, ou seja, foram sequenciadas. A complexidade do SSS é da ordem de $O(n^2k)$, segundo

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

```

1 Inicializar  $e \leftarrow 1, S_e \leftarrow \emptyset$  ;
2 enquanto  $e \leq n$  faça
3   Calcular  $D_e$  e  $\pi R_{kt}$ , para  $t = 1, \dots, T, k = 1, \dots, K^\rho$  ;
4    $j^* \leftarrow \min_{j \in D_e} \{j : v(j) = \min_{i \in D_e} \{v(i)\}\}$  ;
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    $S_e \leftarrow S_e \cup \{j^*\}$  ;
8    $e \leftarrow e + 1$  ;
9 fim

```

Demeulemeester (2002). A Tabela 2.4 apresenta o passo-a-passo da execução do SSS por meio da atribuição de modos apresentada pela Figura 2.2. Nessa tabela, o símbolo “-” (corte) indica que uma tarefa j^* foi removida de D_e e foi incluída em S_e . Já a Figura 2.3 mostra o sequenciamento obtido ao término do SSS e a Figura 2.4 apresenta a codificação completa em listas desse sequenciamento.

Tabela 2.4: Exemplo de execução passo-a-passo do Algoritmo SSS.

e	D_e	j^*	EFT_{j^*}	FT_{j^*}	S_e
1	{1}	1	0	0	{1}
2	{2,3}	2	3	3	{1,2}
3	{3,4}	4	4	4	{1,2,4}
4	{3,6}	6	8	8	{1,2,4,6}
5	{3}	3	2	6	{1,2,4,6,3}
6	{5}	5	8	8	{1,2,4,6,3,5}
7	{7}	7	12	12	{1,2,4,6,3,5,7}
8	{8}	8	12	12	{1,2,4,6,3,5,7,8}

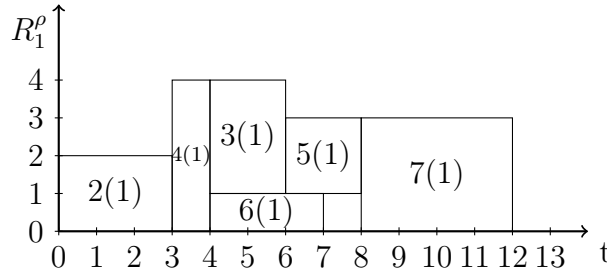


Figura 2.3: Exemplo de sequenciamento obtido pelo SSS a partir da representação dada pela Figura 2.2

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1
L_T	0	3	4	7	6	8	12	12

Figura 2.4: Exemplo de codificação em listas para o problema de projeto representado pela Figura 2.1.

Pela Tabela 2.4, percebe-se que, a cada estágio e , somente uma tarefa é sequenciada (colocada em S_e) e que a ordem da escolha das tarefas, a cada estágio, está de acordo com a lista L_J apresentada pela Figura 2.2. No sequenciamento dado pela Figura 2.3, cada tarefa é representada por $j(m)$, tal que j é a tarefa e m o modo de execução atribuído. O sequenciamento é viável em relação à disponibilidade do recurso não renovável $R_2^\nu = 15$, pois a quantidade requerida pelas tarefas, para os modos atribuídos, é tal que $\sum_{j=1}^8 r_{jm2}^\nu = 15$. O *makespan* do projeto é de 12 unidades de tempo. A disponibilidade do recurso renovável $R_1^\rho = 4$ é respeitada ao longo de todo o horizonte de realização do projeto. Nota-se também que as tarefas fictícias 1 e 8 não aparecem no sequenciamento, pois apresentam duração nula e não consomem quaisquer recursos.

Verificando a Tabela 2.3, percebe-se que as tarefas (reais) críticas 3, 5 e 7 não estão em sua janela $[EST, LFT]$, calculada, respectivamente, como $[0, 2]$, $[2, 4]$ e $[4, 8]$. Esse atraso nessas tarefas acontece pela indisponibilidade de recursos renováveis, dentro das respectivas janelas de tempo, devido à concorrência com outras tarefas nesses mesmos períodos de tempo. Isso criou um atraso no projeto, sendo que este é finalizado com *makespan* maior que o caminho crítico de 8 unidades de tempo.

2.3.2.2 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 n estágios, sendo que, a cada estágio, um conjunto de tarefas é sequenciada. Como descrito em Kolisch (1996b), cada estágio e é associado a um tempo de sequenciamento t_e . Devido a esse tempo, o conjunto de tarefas sequenciadas é dividido nos subconjuntos: (i) Conjunto completo C_e , que representa tarefas que foram sequenciadas e completadas até o tempo de sequenciamento, enquanto o (ii) conjunto ativo A_e representa tarefas que foram sequenciadas, entretanto não completadas, ou seja, estão ativas no tempo de sequenciamento corrente. Finalmente, o já conhecido (iii) conjunto de decisão D_e , que, de acordo com Kolisch (1996b), em contraste com o método serial, D_e contém todas as tarefas não sequenciadas que estão disponíveis para o sequenciamento em relação às restrições de precedência e consumo de recursos.

Sejam os conjuntos A_e e C_e , então πR_{kte} , que é o restante disponível do recurso k no tempo de sequenciamento t_e , e o conjunto D_e , são definidos, respectivamente, como:

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

```

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

$$\begin{aligned} \pi R_{kte} &= R_k^\rho - \sum_{j \in A_e} r_{jmk}^\rho \\ D_e &= \{j : j \notin \{C_e \cup A_e\}, \mathcal{P}_j \subseteq C_e, r_{jmk}^\rho \leq \pi R_{kte}, k = 1, \dots, K^\rho\} \end{aligned}$$

Então, [Kolisch \(1996b\)](#) apresenta a descrição formal do método PSS, mostrada no Algoritmo 3.

Pelo Algoritmo 3, cada estágio é composto de dois passos:

- (i) O novo tempo de sequenciamento t_e é determinado e tarefas com tempo de término igual a t_e são removidas de A_e e colocadas em C_e . Com isso, possivelmente, algumas tarefas serão acrescentadas em D_e ;
- (ii) Uma tarefa $j \in D_e$ é selecionada pelo valor $v(j)$ e sequenciada com início em t_e . Em seguida, essa tarefa é removida de D_e e colocada em A_e .

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

A Tabela 2.5 apresenta o passo-a-passo da execução do Algoritmo 3 por meio da atribuição de modos apresentada pela Figura 2.2.

Tabela 2.5: Exemplo de execução passo-a-passo do Algoritmo PSS

e	t	D_e	j^*	FT_{j^*}	A_e	C_e
1	0	$\{1\}$			$\{\}$	$\{\}$
		$\{1\}$	1	0	$\{1\}$	
2	0	$\{2,3\}$	2	3	$\{2\}$	$\{1\}$
		$\{\}$				
3	3	$\{3,4\}$	4	4	$\{4\}$	$\{1,2\}$
		$\{\}$				
4	4	$\{3,6\}$	6	7	$\{6\}$	$\{1,2,4\}$
		$\{3\}$	3	6	$\{6,3\}$	
5	6	$\{5\}$	5	8	$\{6,5\}$	$\{1,2,4,3\}$
		$\{\}$				
6	7	$\{\}$			$\{5\}$	$\{1,2,4,3,6\}$
7	8	$\{7\}$	7	12	$\{7\}$	$\{1,2,4,3,6,5\}$
		$\{\}$				
8	12	$\{8\}$	8	12	$\{8\}$	$\{1,2,4,3,6,5,7\}$

Pela Tabela 2.5, percebe-se que, a cada estágio e , mais de uma tarefa pode ser sequenciada (colocada em A_e), desde que haja recursos disponíveis no tempo de sequenciamento t_e . A ordem da escolha das tarefas, quando estão no conjunto D_e , a cada estágio, está de acordo com a lista L_J apresentada pela Figura 2.2. Percebe-se, também, as atualizações em D_e a cada estágio. Por exemplo, no início do estágio $e = 3$, $t_e = 3$, $C_e = \{1, 2\}$ e $D_e = \{3, 4\}$. Ao selecionar a tarefa 4, ela é excluída de D_e e é acrescentada em A_e . No segundo momento, a tarefa 3 torna-se inviável para o tempo $t_e = 3$, devido à indisponibilidade de recursos nesse período. Sendo assim, a tarefa 3 também é removida de D_e .

Após a execução do PSS, o mesmo sequenciamento representado pela Figura 2.3 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 tarefas L_J , tal lista já determina a ordem em que as tarefas 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 tarefas pelos valores $v(j)$, $j = 1, \dots, n$, e, então, sequenciar as tarefas, de acordo com a ordem em que aparecem na lista, sem a necessidade de executar as linhas 4 do SSS e 10 do PSS.

Os trabalhos Kolisch e Drexel (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.

2.3.3 Regras de Prioridade

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

Regras de prioridade definem valores de prioridade $v(j)$ para uma tarefa j e para modos de execução m_j , permitindo atribuir modos de execução às tarefas e permitindo aos métodos PSS ou SSS sequenciarem a melhor tarefa j^* , em algum instante de tempo, baseando-se no seu valor de prioridade. A escolha da melhor tarefa 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 tarefas, as regras de prioridade estão relacionadas a tempo e/ou recursos e serão discutidas em seguida.

2.3.3.1 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 2.6 e discutidas em seguida.

Tabela 2.6: 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>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 $

Boctor (1993) foi o primeiro a apresentar regras de prioridade baseadas em modos de execução para a resolução do MRCPSP, 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 tarefas 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 ocorrer 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 tarefas finais que poderiam ser executadas em paralelo com

tarefas 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 tarefas em paralelo, por meio da disponibilidade de recursos. A regra LRP determina a melhor prioridade para o modo com a menor quantidade de consumo relativo, entre todos os tipos de recursos, sendo r_{mk} a quantidade consumida do recurso renovável (ou não renovável) k pelo modo m e R_k a capacidade total do recurso k .

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

Drexler e Gruenewald (1993) apresentam uma regra para a atribuição de modos definida como STOCOM1. Esta regra avalia cada par modo-tarefa, em relação a duração para todos os modos, somente das tarefas elegíveis. Segundo Drexler e Gruenewald (1993), essa regra tem uma tendência de atribuir o modo com menor duração para tarefas 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 tarefas. Buddhakulsomsiri e Kim (2007) utilizaram as regras SFM e LRP, porém adaptaram a última para que fosse considerada a duração de cada tarefa. 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 é gerada, com atribuição aleatória (RAND) de um modo para cada tarefa. 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.

Em Agarwal e Bhat (2013), os autores 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 tarefa 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 tarefa 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 tarefa, 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 tarefas, 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), Bilollikar (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), 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), Bilollikar et al. (2012), Okada et al. (2014), Afshar-Nadjafi (2014), Cui e Yu (2014), Andreica e Chira (2014), Soliman e Elgendi (2014b), Khalilzadeh (2015) e Cheng e Tran (2016) descrevem a atribuição de modos de forma aleatória (RAND) na construção de soluções iniciais.

Como pode ser percebido, a literatura destaca a atribuição de modos de execução para tarefas que levam em consideração a duração das tarefas, com destaque para a regra SFM. Nota-se, também, que vários autores utilizaram 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 no trabalho de Kolisch e Drexel (1996).

2.3.3.2 Regras de prioridade para tarefas

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 tarefas e uma regra de prioridade específica é utilizada, com o objetivo de escolher uma ou mais tarefas para o sequenciamento em construção.

Regras de prioridade para tarefas, no contexto de sequenciamento em projetos, foram aplicadas ao problema RCPSP e adaptadas ao MRCPSp. 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 tarefas ou da escolha da tarefa na construção de um sequenciamento parcial:

- (i) regras baseadas em tarefas: considera informações relacionadas somente às

tarefas, desconsiderando o restante do projeto. A Tabela 2.7 apresenta algumas dessas regras;

- (ii) regras baseadas na rede: somente considera informações contidas na representação do projeto como rede. A Tabela 2.8 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 2.9;
- (iv) regras baseadas em recursos: consideram somente informações provenientes da utilização de recursos. A Tabela 2.10 apresenta algumas dessas regras;
- (v) regras compostas: combinam informações das classificações anteriores. A Tabela 2.11 exemplifica algumas dessas regras.

Tabela 2.7: 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 2.8: 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 2.9: 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 2.10: 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 2.11: Exemplos de Regras de prioridade compostas

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

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

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 os 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 tarefa-modo. Foram verificadas um total de sete regras para tarefa, 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.

Drexel 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 tarefas, dentre elas LFT, LST, EFT, EST, RWK, SPT e TRD, hibridizadas com a metaheurística *Simulated Annealing*, em uma variação multiobjetivo do MRCPSP.

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) usam somente a regra LFT. 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 tarefa-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 deter-

minística 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 Drexel (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), SLK e RAND, ou seja, são as regras que apresentam melhores soluções ou sequenciamentos.

Capítulo 3

Trabalhos Relacionados

Várias abordagens são encontradas na literatura para a resolução do MRCPSP. Os métodos via *branch-and-bound* e *branch-and-cut* se destacam como abordagens exatas na resolução do MRCPSP e têm como principal característica a garantia de otimalidade ao solucionar o problema em discussão. Já as metaheurísticas recebem destaque como técnicas heurísticas para a resolução do MRCPSP, e, como principal característica, possuem a capacidade de encontrar boas soluções em menor tempo, quando comparadas aos métodos exatos. Há, também, as abordagens híbridas conhecidas como matheurísticas, que usam uma abordagem exata em conjunto com uma (meta)heurística, cujo objetivo é combinar os pontos fortes entre as duas abordagens. Neste capítulo, são apresentadas as abordagens exatas presentes na literatura para resolução do MRCPSP. Em seguida, são discutidas as técnicas de matheurísticas e, por fim, os trabalhos baseados em metaheurísticas são comentados.

3.1 Abordagens Exatas

Um algoritmo é dito exato se encontra a solução ótima de um problema, de acordo com condições matemáticas de otimalidade desse mesmo problema. Dentre os algoritmos exatos, são encontrados os algoritmos de enumeração e 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 et al. \(1989\)](#) propuseram um algoritmo B&B conhecido como Árvore de Precedência para uma variação multi-objetivo do MRCPSP. Uma Árvore de Precedência permite uma enumeração sistemática de atividades, atribuindo modos e tempos de início (ou tempos de término) para as atividades. Antes de discutir a abordagem Árvore de Precedência, considere a seguinte definição:

Definição 3.1.1. *Uma atividade j é elegível para ser sequenciada se todas as suas*

predecessoras $i \in \mathcal{P}_j$ já foram finalizadas.

Em um nível l da árvore, uma atividade elegível j_l é identificada e um modo m_{j_l} é 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, repetindo a computação para determinar um tempo de início para outra tarefa elegível, 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 et al. \(1989\)](#), um caminho da raiz até uma folha representa um sequenciamento, respeitando sempre a restrição de precedência. Em seguida, [Sprecher \(1994\)](#) apresentou a abordagem B&B por meio da Árvore de Precedência para o MRCPSP, e [Sprecher e Drexel \(1998\)](#) aprimoraram a abordagem B&B, 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(quais) atividade(s) elegível(eis) sofrerá(ão) atraso(s) em seu(s) tempo(s) de início, ou seja, se 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 Drexel \(1998\)](#) 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 Drexel \(1998\)](#).

[Schnell e Hartl \(2017\)](#) analisaram abordagens exatas para o RCPSP (*Resource Constrained Project Scheduling Problem*) e propuseram extensões dessas abordagens para o MRCPSP, por meio de modelos de Programação por Restrições. Esses autores foram os primeiros a reportar resultados envolvendo programação matemática para a biblioteca pública MMLIB ([Van Peteghem e Vanhoucke, 2014](#)), inclusive apresentando valores ótimos para várias instâncias da MMLIB.

[Changchun et al. \(2018\)](#) usaram uma abordagem de geração de colunas, porém, em um problema de cadeia de suprimentos, uma variante aplicada do MRCPSP, em que diferentes tipos de produtos são produzidos e coordenados, independentemente, por um gerente de recursos e logística que provê diferentes tipos de veículos para transportar os produtos e atender a demanda dos clientes.

Recentemente, Almeida et al. (2019) também discutem uma outra variante do MRCPSP, conhecida como *Multi-skill Resource-constrained Project Scheduling Problem* (MSRCPSP). Duas formulações matemáticas, uma cujo tempo é considerado contínuo e, na segunda, com o tempo discreto, são revisadas e outras duas formulações são propostas.

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 Matheurísticas

Matheurística, pela definição em Boschetti et al. (2009), é um algoritmo para resolver, eficientemente, problemas de otimização ao combinar ambas as técnicas de programação matemática e as (meta)heurísticas. Essa hibridização ocorre em duas categorias, de acordo com Raidl e Puchinger (2008):

- (i) a (meta) heurística controla as chamadas ao procedimento exato;
- (ii) o procedimento exato é caracterizado como *master*, controlando o uso da (meta)heurística em determinados instantes da busca.

A primeira abordagem baseada em uma matheurística para resolução do MRCPSP com minimização do *makespan* foi apresentada por Zhu et al. (2006), que descreve 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. Nesse 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 (AG) para obter soluções iniciais, e, em alguns momentos durante o processo de busca no espaço de soluções, o AG é usado para gerar soluções diversificadas, evitando uma convergência prematura para um ótimo local. Os autores reportaram que determinaram soluções ótimas para as instâncias da maior classe (J30) da biblioteca pública PSPLIB (cf. Kolisch e Sprecher (1997)), melhores do que os anteriormente conhecidos para essas instâncias. Os autores também reportam os valores de tempo computacional na determinação do sequenciamento ótimo, melhores do que os encontrados por Sprecher e Drexler (1998). Entretanto, verificando a diferença da capacidade de processamento em ambos trabalhos, o último apresenta uma abordagem enumerativa mais rápida. Destaca-se também que, o trabalho de Zhu et al. (2006) ainda é considerado o estado-da-arte em relação a métodos exatos e matheurísticas para o MRCPSP.

Gerhards et al. (2017) combinaram o procedimento *Adaptive Large Neighborhood Search* (ALNS) com *Mixed Integer Programming* (MIP). Os resultados apresentados são fortemente competitivos em relação aos resultados da literatura, caracterizando esse trabalho como estado-da-arte.

Esta tese explora uma matheurística baseada no *Local Branching* (LB), que é uma técnica de geração de pseudo-cortes. Esta técnica será discutida no Capítulo 5. A título de observação, associados a esta tese, foram publicados os trabalhos [Fernandes e de Souza \(2018\)](#) e [Fernandes e de Souza \(2021\)](#), que também apresentam resultados de destaque ao usar matheurística para a solução do MRCPSP.

3.3 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. [Drexler e Gruenewald \(1993\)](#) apresentam a abordagem *Regret-Based Biased Random Sampling*, baseada em Regras de prioridade com um sequenciador em serial. Já [Özdamar e Ulusoy \(1994\)](#) propuseram um algoritmo aproximado utilizando um sequenciador em paralelo. [Kolisch e Drexler \(1996\)](#) 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 Drexler \(1997\)](#) 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 \(1996\)](#).

Após os anos 2000, diversas metaheurísticas foram adaptadas para resolução do MRCPSP. [Jozefowska et al. \(2001\)](#) propuseram 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 RCPSP. 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 utilizam 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 suas observações em seus trabalhos anteriores 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 regra de corte 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 \(1996\)](#)). Em [Lova et al. \(2009\)](#), também é implementado um AG, mas é aplicada uma variação da FBI somente em indivíduos viáveis. Em [Bilollikar 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 [Bilollikar \(2014\)](#) 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 tratar soluções inviáveis em relação aos recursos não renováveis. Entretanto, os autores trabalharam no espaço bi-objetivo do problema MRCPPSP, considerando como 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. Outra abordagem multi-objetivo foi proposta em [Kazemi e Tavakkoli-Moghaddam \(2011\)](#), sendo que os autores utilizaram o *Non-dominated Sorting Genetic Algorithm II* (NSGA-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. 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 as “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 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 implementam um procedimento EDA hibridizado 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*.

[Muritiba et al. \(2018\)](#) apresentaram uma abordagem baseada somente em *Path-Relinking*. Os autores apresentam os melhores resultados heurísticos conhecidos em relação a outros trabalhos da literatura, até o presente momento da escrita desta tese, para as classes de problemas das bibliotecas públicas PSPLIB. [Afshar et al. \(2022\)](#) propuseram uma variação do GA, destacando o comportamento da busca lo-

cal sobre o AG, entretanto esses autores resolveram apenas uma classe de problemas da biblioteca PSPLIP. [Chakraborty et al. \(2020\)](#) apresentam uma abordagem baseada na metaheurística *Variable Neighborhood Search* (VNS) combinada com Regras de Prioridade. Em [Machado-Domínguez et al. \(2021\)](#), é apresentado um Algoritmo Memético (MA) hibridizado com VNS. [Ramos et al. \(2022\)](#) apresentam um algoritmo Iterated Local Search (ILS) com a característica de *multi-start*. Os resultados computacionais de [Muritiba et al. \(2018\)](#), para a biblioteca PSPLIB, assim como para a classe de problemas MMLIB, são os melhores reportados, exceto pelos problemas MMLIB+, cujos resultados de [Chakraborty et al. \(2020\)](#) recebem o destaque da literatura.

Durante a evolução desta tese, foi desenvolvida uma estratégia hibridizada entre as metaheurísticas SIA e ALNS, apresentada em [Fernandes e de Souza \(2017\)](#), que obteve resultados significativos para as instâncias da PSPLIB.

A Tabela 3.1 apresenta, resumidamente, algumas das metaheurísticas citadas anteriormente para resolução do MRCPSP, de acordo com autores, métodos, sequenciadores (SGS), uso de busca local (BL) e tipo de recurso considerado. Alguns dos trabalhos citados não estão presentes na Tabela 3.1, pois abordaram a resolução de variações do MRCPSP, como objetivo, atividades ou múltiplos projetos.

Pela Tabela 3.1, verifica-se que a adoção de algoritmos baseados em metaheurísticas para a resolução do MRCPSP são comuns na literatura, assim como a utilização de técnicas de busca local e do SGS Serial. As metaheurísticas recebem destaque, pois apresentam boa capacidade na resolução do MRCPSP em menor tempo, em relação aos métodos exatos discutidos. A tendência de utilização de busca local é evidente, uma vez que ela aprimora as soluções construídas pelas metaheurísticas.

Para finalizar essa seção, destacam-se os trabalhos em [Kolisch e Hartmann \(1999\)](#), [Brucker et al. \(1999\)](#), [Hartmann e Kolisch \(2000\)](#), [Kolisch e Hartmann \(2006\)](#) e [Van Peteghem e Vanhoucke \(2014\)](#), que apresentam estudos de comparações entre algumas dessas metaheurísticas utilizadas na resolução do MRCPSP. Há também uma revisão recente, no momento de escrita desta tese, apresentada em [Hartmann e Briskorn \(2022\)](#), que comenta sobre as abordagens para resolução do MRCPSP e algumas variações desse problema.

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

Autor	Método	SGS	BL	Recursos
Elmaghraby (1977)	CPM	-	-	R
Talbot (1982)	B&B truncado	Serial	-	R/NR
Drexel e Gruenewald (1993)	Multi-pass	Serial	-	R/NR
Boctor (1993)	Single-pass	Paralelo	-	R
Özdamar e Ulusoy (1994)	Multi-pass	Paralelo	-	R/NR
Kolisch e Drexel (1996)	Multi-pass	Serial/Paralelo	-	R/NR
Boctor (1996)	Multi-pass	Paralelo	-	R
Kolisch e Drexel (1997)	Multi-pass	Serial/Paralelo	Sim	R/NR
Knotts et al. (2000)	Multi-agentes	-	-	R
Jozefowska et al. (2001)	SA	Serial	Sim	R/NR
Hartmann (2001)	AG	Serial	Sim	R/NR
Bouleimen e Lecocq (2003)	SA	Serial	Sim	R/NR
Alcaraz et al. (2003)	AG/Multi-pass	Serial	Sim	R/NR
Zhang et al. (2006)	PSO	Serial	Sim	R/NR
Jedrzejowicz e Ratajczak (2006)	Aprendizado	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	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	R/NR
Van Peteghem e Vanhoucke (2010)	AG/FBI	Serial	Sim	R/NR
Elloumi e Fortemps (2010)	AG	Serial	Sim	R/NR
Muller (2011)	ALNS	Serial	Sim	R/NR
Van Peteghem e Vanhoucke (2011)	<i>Scatter-search</i>	Serial	Sim	R/NR
Wauters et al. (2011)	Multi-agentes	Serial	Sim	R/NR
Wang e Fang (2011)	SFLA	Serial	Sim	R/NR
Khalilzadeh et al. (2012)	PSO	Paralelo	Sim	R/NR
Bilollikar et al. (2012)	AG/Multi-pass	Serial	Sim	R/NR
Wang e Fang (2012)	EDA	Serial	Sim	R/NR
Atli e Kahraman (2012)	BT	-	Sim	R/NR
Li e Zhang (2013)	ACF	Serial	-	R/NR
Agarwal e Bhat (2013)	AE	-	-	R/NR
Mirzaei e Akbarzadeh-T (2013)	Multi-agentes	Serial	-	R/NR
Afshar-Nadjafi (2014)	SA	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	Serial	-	R/NR
Cheng e Tran (2016)	ED/C-Means	Serial	-	R/NR
Fernandes e de Souza (2017)	SIA/ALNS	Serial	Sim	R/NR
Fernandes e de Souza (2018)	LB	Serial	Sim	R/NR
Muritiba et al. (2018)	<i>Path Relinking</i>	Serial	Sim	R/NR
Afshar et al. (2022)	GA	Serial	Sim	R/NR
Chakraborty et al. (2020)	VNS	Serial	Sim	R/NR
Machado-Domínguez et al. (2021)	MA	Serial	Sim	R/NR
Fernandes e de Souza (2021)	LB	Serial	Sim	R/NR
Ramos et al. (2022)	M-ILS	Serial	Sim	R/NR

Capítulo 4

Referencial Teórico

Neste capítulo é discutida a fundamentação teórica para a tese. São abordados os termos e conceitos usados nas estratégias *Branch-and-Bound* e *Branch-and-Cut*, bem como é detalhada a estratégia *Local Branching*.

4.1 *Branch-and-Bound* e *Branch-and-Cut*

Como descrito em [Bertsimas e Tsitsiklis \(1997\)](#) e [Papadimitriou e Steiglitz \(1998\)](#), um método baseado em *Branch-and-Bound* (B&B) utiliza uma abordagem de “divisão e conquista” para explorar o espaço de soluções inteiras. Essa abordagem utiliza limites (*bounds*) para evitar a exploração de algumas áreas do espaço de soluções que não contém uma solução inteira melhor do que a solução inteira corrente.

Os algoritmos baseados em *Branch-and-Bound* (B&B) discutidos no Capítulo 3 constroem um sequenciamento, criando uma árvore de enumeração do B&B à medida que se sequencia cada tarefa, até que todas estas estejam sequenciadas. Nesta seção, o método B&B será discutido no contexto de PLI.

Seja F o conjunto de soluções viáveis de um problema de PLI, na forma:

$$\min\{c'x : Ax \leq b, \ x \geq 0, \ x \text{ é inteiro}\}, \quad (4.1)$$

O conjunto F é particionado em uma coleção finita de subconjuntos (ou subproblemas) $F_1, F_2, \dots, F_m$, que são resolvidos separadamente. As soluções ótimas de cada subproblema F_i são comparadas e a melhor é escolhida. Durante essas divisões, alguns subproblemas podem ser resolvidos de forma ótima, ou pelo menos uma solução viável é encontrada. Essas soluções apresentam um limite superior, que refere-se à melhor solução viável encontrada até um certo instante. Como cada subproblema pode ser tão difícil de resolver quanto o problema original, este também é dividido. Essa divisão é chamada de *branching*, que gera uma árvore de subproblemas, conforme exposto na Figura 4.1.

Algoritmo 4: Esquema de *Branch-and-Bound* – Problema de minimização

Entrada: $\bar{x}, Q = \{F\}$

```

1 //  $\bar{x}$  é a solução incumbente ;
2 //  $Q$  é a fila de subproblemas ;
3 enquanto  $Q \neq \emptyset$  faça
4    $N =$  selecionar um subproblema em  $Q$  ;
5    $Q = Q - \{N\}$  ;
6   branch em  $N$  gerando  $F_1, F_2, \dots, F_m$  ;
7   para  $i = 1, 2, 3, \dots, m$  faça
8      $x^* = \text{solucionar}(F_i)$  // bound ;
9     se  $x^*$  é ótima ou viável AND  $f(x^*) < f(\bar{x})$  então
10       $\bar{x} = x^*$  ;
11      continue ;
12   fim
13   se  $lb(F_i) < f(\bar{x})$  então
14      $Q = Q \cup \{F_i\}$  ;
15   fim
16 fim
17 fim
18 return  $\bar{x}$  // solução ótima;
```

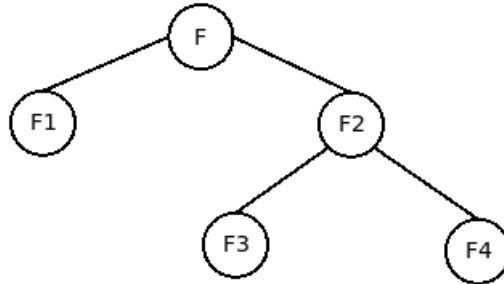


Figura 4.1: Uma árvore de subproblemas. O conjunto inicial F é particionado em F_1 e F_2 . F_1 não será explorado, pois apresenta um limite inferior maior do que a melhor solução conhecida. Em seguida, F_2 é particionado em F_3 e F_4 .

De acordo com [Bertsimas e Tsitsiklis \(1997\)](#), deve existir algum algoritmo eficiente que, para cada $F_i, i = 1, \dots, m$, computa um limite inferior em relação ao custo ótimo do subproblema corrente. Uma maneira de obter os limites é usar a solução ótima de um problema de Programação Linear Relaxado (PLR), tal como

$$\min\{c'x : Ax \leq b, x \geq 0\}. \quad (4.2)$$

Sendo assim, o esquema de B&B é formalizado pelo Algoritmo 4 e é resumido pelos seguintes passos:

Passo 1: Selecionar um subproblema F_i ;

Passo 2: Se F_i é inviável, então será deletado. Caso contrário, um limite inferior $lb(F_i)$ de F_i é calculado;

Passo 3: Se $lb(F_i) \geq U$, para U sendo algum limite superior conhecido, então F_i será deletado.

Passo 4: Se $lb(F_i) < U$, então uma solução ótima do subproblema pode ser procurada ou o subproblema é particionado em outros subproblemas.

O método *Branch-and-Cut* (B&C) é uma variação do método *Branch-and-Bound*, que utiliza cortes (desigualdades lineares) para resolver os subproblemas. As desigualdades lineares são adicionadas à formulação, gerando, então, os subproblemas, com o objetivo de aprimorar os limites obtidos por algum algoritmo eficiente. Para Lodi (2009), abordagens baseadas em geração de cortes vêm sendo continuamente usadas e a principal discussão é sobre como escolher os cortes gerados, já que o desempenho de uma abordagem de geração de cortes depende da qualidade da desigualdade acrescentada.

4.2 Matheurísticas

Como exposto em Boschetti et al. (2009), o campo das Matheurísticas tem recebido destaque na literatura, cujo objetivo está em como estruturar um algoritmo para solucionar de forma eficiente um problema de otimização. A Matheurística é um algoritmo híbrido entre (meta)heurísticas e programação matemática. A hibridização, de acordo com Boschetti et al. (2009), acontece de duas maneiras:

- (i) a (meta)heurística controla as chamadas à abordagem exata; ou
- (ii) a abordagem exata é caracterizada como “mestre”, controlando o uso da (meta)-heurística.

Nesta tese, será proposto um esquema de matheurística segundo a metodologia (ii). Isso significa que a abordagem exata, baseada em uma estratégia de geração de pseudo-cortes, controla as chamadas à (meta)heurística. Essa última é representada por um *solver* de programação inteira, que busca a otimalidade em um subconjunto de soluções do espaço de soluções com tempo limite para a busca.

4.2.1 A Estratégia *Local Branching*

As técnicas de programação matemática discutidas nas seções anteriores deste Capítulo se apresentam na literatura como propostas de resolução do MRCPSP, como foi detalhado no Capítulo 3. Entretanto, usar puramente essas abordagens, de acordo com Lodi (2009), não é a melhor maneira de resolver problemas de sequenciamento. Para Lodi (2009), uma boa questão é saber se existe um bom modelo de Programação Linear Inteira para um problema de sequenciamento ou, então, os pesquisadores devem atentar-se a estender o modelo do problema para diferentes direções, como, por exemplo, a hibridização.

Seja (PLI) um problema de Programação Linear Inteira:

$$(PLI) \quad \min \quad c'x \quad (4.3)$$

$$\text{s.a.} \quad Ax \geq b \quad (4.4)$$

$$x_j \geq 0, \quad j \in \mathbb{N} \quad (4.5)$$

$$x_j \in \{0, 1\}, \quad j \in \mathbb{I}, \quad (4.6)$$

tendo, associada, uma solução viável de referência $\bar{x}$ de (PLI) .

Fischetti e Lodi (2003) apresentaram o método *Local Branching* (LB) para a solução de PLIs baseado na fixação de variáveis, com o objetivo de garantir a otimalidade na vizinhança de $\bar{x}$. Eles discutem como definir uma vizinhança para uma solução viável de referência e, também, a possibilidade de gerar cortes e/ou pseudo-cortes, para excluir partes do espaço de soluções.

A estratégia LB fixa algumas variáveis, isto é, exige que uma certa quantidade de variáveis assumam os mesmos valores que a solução de referência, sem fixar as outras variáveis. Sendo assim, tem-se a adição de uma restrição linear ao problema original. A restrição adicionada:

$$\delta(x, \bar{x}) \leq kc \quad (4.7)$$

define, de acordo com Fischetti e Lodi (2003), uma vizinhança k -ótima N^k de $\bar{x}$.

A vizinhança N^k , para algum k inteiro, é então definida como:

$$N^k(\bar{x}) = \{x \text{ obtida de } (PLI) : \delta(x, \bar{x}) \leq k\} \quad (4.8)$$

Como discutido em Fischetti e Lodi (2003), a Expressão (4.7) pode ser usada com uma regra de *branching* para (PLI) . Essa regra é uma desigualdade linear, caracterizada como pseudo-corte, e gerada com o propósito de excluir certas áreas do espaço de busca, evitando soluções que poderiam ser aceitas como candidatas de um problema de otimização.

O uso de pseudo-cortes foi inicialmente sugerido em Glover (1990), quando o autor discute a estratégia de Busca Tabu. Em uma de suas variações, o pseudo-corte é apresentado nos movimentos chamados pelo autor de *movimento de relaxação* e *movimento de complementação*, e são usados como uma restrição temporária durante o processo de busca.

O prefixo *pseudo* refere-se ao fato da desigualdade *poder* não ser válida no sentido de não garantir que pelo menos um ótimo global estará no subproblema a explorar, isto é, não há uma condição matemática suficiente para garantir esse ótimo global no subproblema gerado pela desigualdade. Por isso, essas desigualdades também são conhecidas como não válidas. Entretanto, podem ser usadas em um esquema de *branching*.

Dado $\bar{x}$, o espaço de soluções associado ao subproblema atual em algum esquema de enumeração pode ser particionado pelas regras:

$$\delta(x, \bar{x}) \leq k \text{ (left branch)} \quad \text{ou} \quad \delta(x, \bar{x}) \geq k + 1 \text{ (right branch)} \quad (4.9)$$

Seja (P) a formulação do MRCPSP apresentada na Seção 2.2, $\bar{x}$ a solução factível inicial construída e X o conjunto de todas as soluções factíveis de (P) . Então, a busca em uma vizinhança $N^k(\bar{x})$ pode ser definida adicionando o pseudo-

corde $\delta(x, \bar{x}) \leq k$; já o novo conjunto de soluções $X = X \setminus N^k(\bar{x})$ é representado eliminando-se $\delta(x, \bar{x}) \leq k$ e adicionando o pseudo-corde $\delta(x, \bar{x}) \geq k + 1$.

Proposição 4.2.1. *O pseudo-corde $\delta(x, \bar{x}) \geq k+1$ garante que o subproblema gerado por $\delta(x, \bar{x}) \leq k$ não seja revisitado durante o restante da busca.*

Proposição 4.2.2. *$\delta(x, \bar{x}) \geq k + 1$ não determina uma perda da informação sobre a solução ótima global para o problema (P) original durante a redução do espaço de soluções.*

Demonstração. A afirmação na Proposição (4.2.1) é simples e óbvia, uma vez que as regiões geradas pelo *left-branch* e *right-branch* são mutuamente excludentes para a mesma solução de referência. Em relação à afirmação da Proposição (4.2.2), seja X o conjunto de todas as soluções factíveis de (P), x^* a solução ótima global e $\bar{x}$ a solução de referência. Inicialmente, $x^* \in X$. Assume-se, por contradição, que a inserção de $X \cap \delta(x, \bar{x}) \geq k + 1$ determina a perda da informação sobre o ótimo global de X . Ao inserir o pseudo-corde $\delta(x, \bar{x}) \leq k$, cria-se um subproblema Y a ser otimizado. Há então duas possibilidades: (i) x^* não está em Y ou (ii) x^* está em Y . Para o caso (i), após a exclusão de $\delta(x, \bar{x}) \leq k$ para inclusão permanente de $X \cap \delta(x, \bar{x}) \geq k + 1$, fica claro que x^* está em $X \cap \delta(x, \bar{x}) \geq k + 1$. Para o caso (ii), após deletar o pseudo-corde $\delta(x, \bar{x}) \leq k$, a informação sobre o ótimo global permanece, uma vez que ele passa a ser a nova solução de referência, pois Y foi otimizado. Então, há a inclusão de $\delta(x, \bar{x}) \geq k + 1$. Mas, nesse momento, $x^* = \bar{x}$ e a informação sobre o ótimo global permanece conhecida para o processo de busca. Em ambos os casos gerou-se uma contradição, uma vez que foi assumido, por hipótese, que a inserção de $X \cap \delta(x, \bar{x}) \geq k + 1$ determina a perda da informação com relação ao ótimo global de X . $\square$

O parâmetro k na Expressão (4.7) deve ser escolhido de forma apropriada para garantir um subproblema mais fácil de resolver após a adição do *left branch* em P , isto é, a vizinhança $N^k(\bar{x})$ deve ser suficientemente pequena para ser otimizada em tempo aceitável, bem como grande o suficiente na tentativa de encontrar soluções melhores que $\bar{x}$. Essa abordagem do LB descrita até o momento corresponde a uma variação do método exato *Branch-and-Cut*, utilizando pseudo-cortes. A conversão do LB para uma heurística pode ser descrita ao adicionar o parâmetro `node_time_limit`, que é usado para incorporar um tempo limite para a exploração do subproblema *left branch*. Esse parâmetro define o tempo limite para resolver cada subproblema gerado pela restrição *left branch*.

Uma representação da estratégia LB é apresentada na Figura 4.2. No início, há uma solução de referência $\bar{x}_1$ para o nó “1”, que contempla todo o modelo do MRCPSp apresentado no Capítulo 2. Então, o pseudo-corde linear $\delta(x, \bar{x}_1) \leq k$ deve ser adicionado, gerando, assim, o subproblema “2”, que é resolvido de forma ótima (destacado pelo símbolo “*”), encontrando a nova solução de referência $\bar{x}_2$. Em seguida, $\delta(x, \bar{x}_1) \leq k$ é retirado para a adição permanente da desigualdade $\delta(x, \bar{x}_1) \geq k + 1$. Ao adicionar o pseudo-corde $\delta(x, \bar{x}_2) \leq k$, o subproblema “4” é resolvido, encontrando-se uma solução viável de melhora $\bar{x}_3$. Como esse último subproblema não foi resolvido na otimalidade (destacado pelo símbolo “+”), o pseudo-corde anterior é removido, para adição de $\delta(x, \bar{x}_3) \leq k$, que gera o novo subproblema “5”, solucionado de forma ótima, obtendo então a nova solução de referência $\bar{x}_4$.

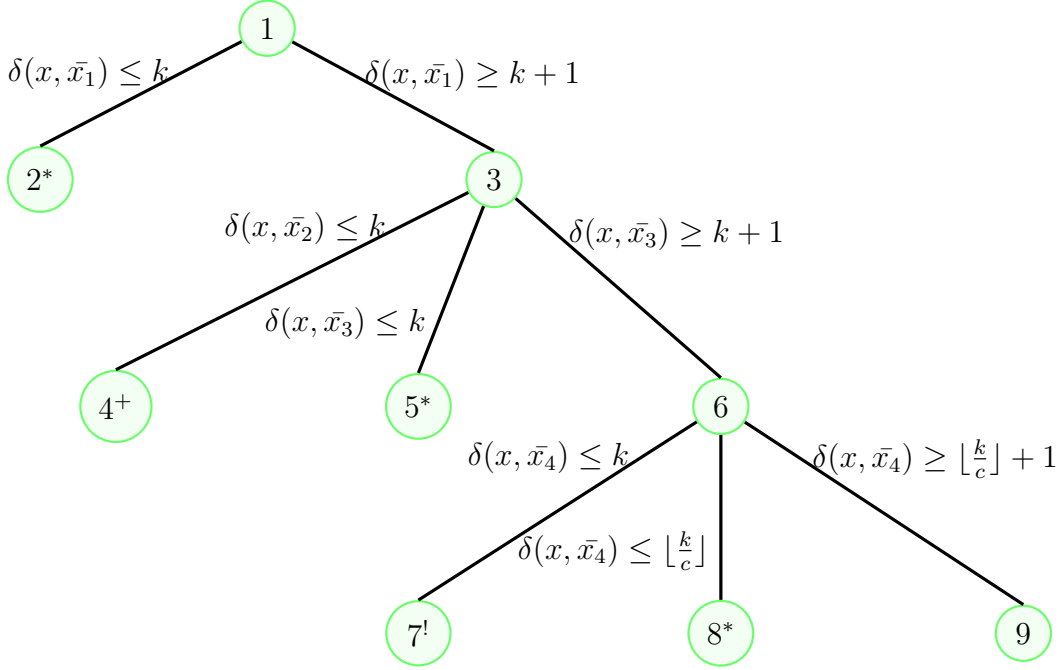


Figura 4.2: Esquema de enumeração da estratégia LB. Adaptado de Fischetti e Lodi (2003).

Assim, o pseudo-corte anterior é removido para a adição de mais um pseudo-corte permanente, $\delta(x, \bar{x}_3) \geq k + 1$. É importante destacar que, até esse instante, foram adicionados 2 pseudo-cortes ao modelo do MRCPS. No subproblema “7”, o *PLI solver* terminou sua execução devido ao parâmetro `node_time_limit`, sem encontrar uma solução de melhora (destacado pelo símbolo “!”). Nesse instante, deverá acontecer uma diminuição da vizinhança, controlada pelo parâmetro k , o que provoca a adição de $\delta(x, \bar{x}_2) \leq \frac{k}{c}$, para algum valor positivo c , gerando o subproblema “8”. Este último subproblema é resolvido de forma ótima, o que implica na adição, permanente, do pseudo-corte $\delta(x, \bar{x}_4) \geq \frac{k}{c} + 1$, e, assim, a estratégia LB continua a explorar o espaço de soluções.

Fischetti e Lodi (2003) apresentam o LB de acordo com o Algoritmo 5. Esse algoritmo está de acordo com o processo de enumeração descrito no Capítulo 3. Para o valor inicial do parâmetro k , Fischetti e Lodi (2003) sugerem $k_0 = 20$.

Seja X o conjunto de todas as soluções viáveis de (P) . Então, a busca em uma vizinhança $N^k(\bar{x})$ é definida assim que o pseudo-corte $\delta(x, \bar{x}) \leq k$ é adicionado em (P) . O novo conjunto de soluções $X = X \setminus N^k(\bar{x})$ é definido ao excluir $\delta(x, \bar{x}) \leq k$ de (P) e adicionar $\delta(x, \bar{x}) \geq k + 1$.

A busca descrita no Algoritmo 5 inicia com uma solução viável $\bar{x}$ de (P) . Após a adição do *left branch*, é criado um subproblema que será processado por um *MIP solver*, conforme a linha 4. Dessa maneira, há três possibilidades para o resultado retornado pelo *MIP solver*:

- (i) Caso 1 (linhas 5-10): Se o *MIP solver* encontra uma solução x^* ótima ou viável, que seja melhor do que a solução de referência, então, há um *backtrack* ao nó anterior, retirando-se o *left branch* e adicionando o *right branch* em (P) . Esse passo é representado, simplesmente, na linha 7. Por fim, atualiza-se a solução

Algoritmo 5: Local Branching - (LB)

Entrada: $\bar{x}$, k_0 , node_time_limit

```

1  $k = k_0$  ;
2  $k_{\min} = 1$  ;
3 enquanto critério de parada não satisfeito faça
4    $x^* = \text{MIP\_SOLVER}(P, X \cap N^k(\bar{x}), \text{node\_time\_limit})$  ;
5   se ( $x^* \in \{X \cap N^k(\bar{x})\}$  é ótimo ou viável) AND ( $x^*$  é melhor que  $\bar{x}$ ) então
6     se  $x^*$  é ótimo então
7        $X = X \setminus N^k(\bar{x})$  ;
8       reset  $k$  e  $k_{\min}$  ;
9     fim
10     $\bar{x} = x^*$  ;
11  senão
12    se (nenhuma solução viável  $x^* \in \{X \cap N^k(\bar{x})\}$  encontrada) OR ( $x^*$  é ótima e
13      não é melhor que  $\bar{x}$ ) então
14       $k_{\min} = k$  ;
15       $k = k + \text{round}(k/2)$  ;
16    senão
17       $k = k - \text{round}(k/2)$  ;
18      se  $k \leq k_{\min}$  então
19         $\bar{x} = \text{SimpleStrongDiversification}(\bar{x})$  ;
20      fim
21    fim
22 fim
23 return  $\bar{x}$ 

```

de referência para x^* . Caso x^* não seja ótima, mas é uma solução melhor do que a solução de referência, então, há apenas, a atualização para a nova solução de referência e a troca entre os pseudo-cortes *left branch*;

- (ii) Caso 2 (linhas 12-14): Se o *MIP solver* encontra uma solução ótima x^* que não é melhor do que $\bar{x}$, ou nenhuma solução viável é retornada, então a vizinhança $N^k(\bar{x})$ precisa aumentar. Para isso, aumenta-se o valor de k , e, consequentemente, o método LB faz a busca em um subproblema de maior cardinalidade. É importante dizer que há apenas a troca entre os pseudo-cortes *left branch* e não há a adição de um *right branch*, uma vez que não há critérios matemáticos para excluir uma região do espaço de busca;
- (iii) Case 3 (linha 16): Se o tempo limite de busca em um subproblema é alcançado, então é necessário reduzir a cardinalidade da vizinhança para uma busca em um subproblema menor. Para isso, reduz-se o valor de k .

Quando a cardinalidade da vizinhança é diminuída por meio de k , há a possibilidade do Algoritmo 5 entrar em um ciclo, para valores de k iguais (ou menores) que $k_{\min}$. Caso isso aconteça, é necessário um procedimento de perturbação para diversificação. Fischetti e Lodi (2003) destacam essa ação como *Strong Diversification*. Esse procedimento de perturbação é necessário para retirar a matheurística

do ótimo local, gerando uma nova solução de referência. Essa modificação altera a solução corrente $\bar{x}$ para obter uma nova solução de referência $\bar{x}$.

Capítulo 5

Metodologia

Neste capítulo são descritos todos os algoritmos propostos nesta tese. Será apresentada uma variante do método *Local Branching* para resolução do MRCPSP. Também são caracterizados os pseudo-cortes para uso no processo de enumeração da variante LB. Finalmente, são discutidos os procedimentos heurísticos de perturbação e busca local presentes na variante proposta do método LB.

5.1 Esquema de Matheurística Proposto

Nesta tese, o esquema de matheurística proposto *Local Branching* (LB), para a resolução do MRCPSP, controla as chamadas a um *solver* para Programação Linear Inteira (PLI) e, assim, explora o espaço de soluções. Também serão propostas duas novas caracterizações de pseudo-cortes para o MRCPSP, bem como outras duas caracterizações, presentes na literatura, serão discutidas. As (meta)heurísticas que foram usadas como complemento para o LB também serão discutidas.

O Algoritmo 6 apresenta a adaptação do método *Local Branching* para resolução do MRCPSP. Há algumas diferenças entre a variação proposta nesta tese e a versão original do LB, mostrada no Algoritmo 5. São elas:

- O valor k_{min} representa a vizinhança de maior cardinalidade, que já foi verificada completamente. Por exemplo, quando $k_{min} = 4$, então a vizinhança de cardinalidade $|N^k(\bar{x})| \leq 4$ já foi completamente verificada e não houve solução melhor do que a solução corrente. Já o valor k_{aux} representa a menor cardinalidade para uma vizinhança que não foi completamente verificada; assim, por exemplo, quando $k_{aux} = 10$, então a vizinhança $|N^k(\bar{x})| \leq 10$ não foi totalmente verificada. Esses valores melhoram o método LB, evitando o processamento desnecessário de uma parcela de soluções já processada completamente;
- O parâmetro β é usado para diversificação, que será detalhado mais a frente.
- Observe que o parâmetro k aumenta e diminui de forma mais rápida, de acordo com as linhas 19 e 26 do Algoritmo 6. O incremento ou decremento na cardinalidade da vizinhança trouxe destaque para essa variação, em relação à versão original, assim como o valor inicial de k .

- O parâmetro k deve ser par. A demonstração desta afirmação está na Seção 5.2.

De acordo com o Algoritmo 6, duas perguntas surgem:

- Como são geradas as vizinhanças (subproblemas) $N^k(\bar{x})$?
- Como é realizada a perturbação em $\bar{x}$?

As respostas para as perguntas (i) e (ii) são detalhadas nas seções seguintes.

5.2 Local Branching Cut

A classe de desigualdades para gerar os subproblemas (as vizinhanças) é conhecida como *Local Branching Cut* e foi proposta por Fischetti e Lodi (2003). Essa classe de desigualdades define um pseudo-corte por meio de um conjunto suporte $\bar{S}$ em relação a uma solução viável de referência $\bar{x}$. Sendo assim, definindo-se $\bar{S}$, caracteriza-se um subproblema (vizinhança) para o *Local Branching Cut*.

Fischetti e Lodi (2003) apresentam a desigualdade linear *Local Branching Cut* para gerar os subproblemas como:

$$\delta(x, \bar{x}) = \sum_{j \in \bar{S}} (1 - x_j) + \sum_{j \notin \bar{S}} x_j \leq k \quad (5.1)$$

Então, por meio do pseudo-corte linear da Expressão (5.1), é gerada uma vizinhança $N^k(\bar{x})$, cujo grau de diferença entre duas soluções é no máximo k . A Expressão (5.1) é reescrita para o MRCPSP da seguinte maneira:

$$\delta(x_{jmt}, \bar{x}_{j\bar{m}\bar{t}}) = \sum_{j=1}^n \sum_{(m,t) \in \bar{S}_j} (1 - x_{jmt}) + \sum_{j=1}^n \sum_{(m,t) \notin \bar{S}_j} x_{jmt} \leq k, \quad (5.2)$$

ou

$$\delta(x_{jmt}, \bar{x}_{j\bar{m}\bar{t}}) = \sum_{j=1}^n \sum_{t \in \bar{S}_j} (1 - x_{j\bar{m}t}) + \sum_{j=1}^n \sum_{t \notin \bar{S}_j} x_{j\bar{m}t} \leq k \quad (5.3)$$

Essas duas caracterizações são melhor detalhadas na definição de $\bar{S}$.

Após a definição da desigualdade linear que realiza o *branching* durante o processo de busca, algumas vizinhanças são caracterizadas ao se definir $\bar{S}$ e serão discutidas nesta tese. Fischetti e Lodi (2003) definem $\bar{S}$ como sendo:

$$\bar{S} = \{j : \bar{x}_j = 1\}, \quad (5.4)$$

isto é, a vizinhança de uma solução de referência $\bar{x}$, ao definir $\bar{S}$, é caracterizada pelas soluções que se diferem de $\bar{x}$. Essa diferença é permitida até um grau k , conforme discutem Fischetti e Lodi (2003). Percebe-se, então, que o pseudo-corte apresentado na Expressão (5.2) irá criar um subproblema que aceitará soluções com no máximo grau k de diferença em relação à solução de referência $\bar{x}$. Porém, ao se definir $\bar{S}$, há algumas maneiras distintas de considerar diferenças entre duas soluções. Como o MRCPSP possui três índices para a variável binária, a definição de Fischetti e Lodi

Algoritmo 6: Esquema de matheurística baseada em Local Branching

Entrada: $\bar{x}$, k_0 , β , `node_time_limit`

```

1   $k = k_0$  ;
2   $k_{aux} = k_0$  ;
3   $k_{min} = 2$  ;
4  enquanto critério de parada não satisfeito faça
5       $x^* = \text{MIP\_SOLVER}(P, X \cap N^k(\bar{x}), \text{node\_time\_limit})$  ;
6      se ( $x^* \in X \cap N^k(\bar{x})$  é ótima ou viável) AND ( $x^*$  é melhor que  $\bar{x}$ ) então
7          se  $x^*$  é ótima então
8               $X = X \setminus N^k(\bar{x})$  ;
9              reset  $k$ ,  $k_{aux}$ ,  $k_{min}$  ;
10         fim
11          $\bar{x} = x^*$  ;
12     senão
13         se (nenhuma solução viável  $x^* \in X \cap N^k(\bar{x})$  encontrada) OR
14             ( $x^*$  é ótima e não é melhor que  $\bar{x}$ ) então
15              $k_{min} = k$  ;
16             se  $k_{aux} > k$  então
17                  $k = k_{aux}$  ;
18             senão
19                  $k = k * \max(2, \lceil k/2 \rceil)$  ;
20                 se  $k \bmod 2 \neq 0$  então
21                      $k = k + 1$  ;
22                 fim
23             fim
24         senão
25              $k_{aux} = k$  ;
26              $k = \max(2, \lfloor (k + k_{min})/2 \rfloor)$  ;
27             se  $k \bmod 2 \neq 0$  então
28                  $k = k - 1$  ;
29             fim
30             se  $k \leq k_{min}$  então
31                 reset  $k$ ,  $k_{aux}$ ,  $k_{min}$  ;
32                  $\bar{x}' = \text{StrongDiversification}(\bar{x}, \beta, n)$  ;
33                 se  $\bar{x}'$  igual a  $\bar{x}$  então
34                      $\bar{x}'' = \text{StrongDiversificationAtRandom}(n)$  ;
35                     se  $\bar{x}''$  não é null então
36                          $\bar{x}' = \bar{x}''$  ;
37                     fim
38                 fim
39                  $\bar{x} = \bar{x}'$  ;
40             fim
41         fim
42     fim
43 fim
44 return  $\bar{x}$  ;

```

(2003) para caracterizar $\bar{S}$ será melhor explorada nesta tese. Sendo assim, duas caracterizações para os subproblemas são apresentadas e discutidas nesta tese. Pelo estudo da literatura, essas duas caracterizações são novas na literatura. Em seguida, duas outras caracterizações, já existentes na literatura, também são apresentadas e discutidas.

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1
L_T	0	3	4	7	6	8	12	12

(a)

L_J	1	3	2	5	4	6	7	8
L_M	1	1	1	1	1	1	1	1
L_T	0	2	5	4	6	9	10	10

(b)

Figura 5.1: (a) Solução de referência $\bar{x}$ codificada em listas. (b) Solução $x^1 \in N^{16}(\bar{x})$ codificada em listas

5.2.1 Proposta de Caracterização dos Subproblemas

A seguir, são apresentadas duas caracterizações para os subproblemas, a primeira baseada em tempos de término e a segunda baseada em modos e tempos de término.

5.2.1.1 Vizinhança baseada em Tempos de Término

Uma vizinhança para definir o pseudo-corte da Expressão (5.3) é definida ao permitir grau de diferença que envolva somente os tempos de término, isto é, definir $\bar{S}$ como:

$$\bar{S}_j = \{t : \bar{x}_{jmt} = 1, EFT_j \leq t \leq LFT_j\}, \quad j = 1, \dots, n \quad (5.5)$$

Pela Expressão (5.5), para o pseudo-corte da Expressão (5.3), a vizinhança de uma solução de referência $\bar{x}$ é caracterizada pelas soluções que se diferem de $\bar{x}$ em relação a, apenas, diferentes tempos de término das tarefas. Essa caracterização para $\bar{S}$ é restrita, já que, na definição de N^k pelo *left branch*, não é permitida uma solução com modo de execução distinto da solução de referência, o que reduz o MRCPPSP para o RCPSP quando N^k é otimizada.

Como exemplo, seja a solução de referência $\bar{x}$ codificada em listas e apresentada na Figura 5.1(a) e a solução x^1 como uma das soluções vizinhas de $\bar{x}$, apresentada na Figura 5.1(b). Essas soluções são interpretadas como

$$\bar{x} : \bar{x}_{1,1,0} = 1, \bar{x}_{2,1,3} = 1, \bar{x}_{3,1,6} = 1, \bar{x}_{4,1,4} = 1, \bar{x}_{5,1,8} = 1, \bar{x}_{6,1,7} = 1, \bar{x}_{7,1,12} = 1, \bar{x}_{8,1,12} = 1$$

$$x^1 : x_{1,1,0} = 1, x_{2,1,5} = 1, x_{3,1,2} = 1, x_{4,1,6} = 1, x_{5,1,4} = 1, x_{6,1,9} = 1, x_{7,1,10} = 1, x_{8,1,10} = 1$$

Nota-se que $\delta(x, \bar{x}) = 14$, isto é, o grau de diferença é igual a 14. É preciso atentar-se que a mudança no tempo de término da tarefa real que termina o mais tarde no projeto indica, necessariamente, a mudança no tempo da tarefa fictícia final. O leitor pode perceber que nesse exemplo os modos das soluções x^1 e $\bar{x}$ devem ser iguais.

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1
L_T	0	3	4	7	6	8	12	12

(a)

L_J	1	3	2	5	4	6	7	8
L_M	1	1	2	1	1	1	1	1
L_T	0	2	4	4	5	8	9	9

(b)

Figura 5.2: (a) Solução de referência $\bar{x}$ codificada em listas. (b) Solução $x^2 \in N^{16}(\bar{x})$ codificada em listas

5.2.1.2 Vizinhaça baseada em Modos e Tempos de Término

Uma outra vizinhaça para definir $\bar{S}$, em relação ao pseudo-corte da Expressão (5.2), é caracterizada como:

$$\bar{S}_j = \{(m, t) : \bar{x}_{jmt} = 1, EFT_j \leq t \leq LFT_j\}, j = 1, \dots, n \quad (5.6)$$

Ao definir $\bar{S}$ como mostrado pela Expressão (5.6), a vizinhaça de uma solução de referência $\bar{x}$ é caracterizada pelas soluções que diferem-se de $\bar{x}$ em relação a diferentes tempos de término das tarefas e modos de execução, o que torna essa última caracterização mais extensa do que a primeira.

Como exemplo, seja a solução de referência $\bar{x}$ codificada em listas apresentada na Figura 5.2(a). Uma solução vizinha de $\bar{x}$ é a solução x^2 apresentada na Figura 5.2(b). Essas soluções são interpretadas como:

$$\bar{x} : \bar{x}_{1,1,0} = 1, \bar{x}_{2,1,3} = 1, \bar{x}_{3,1,6} = 1, \bar{x}_{4,1,4} = 1, \bar{x}_{5,1,8} = 1, \bar{x}_{6,1,7} = 1, \bar{x}_{7,1,12} = 1, \bar{x}_{8,1,12} = 1$$

$$x^2 : x_{1,1,0} = 1, x_{2,2,4} = 1, x_{3,1,2} = 1, x_{4,1,5} = 1, x_{5,1,4} = 1, x_{6,1,8} = 1, x_{7,1,9} = 1, x_{8,1,9} = 1$$

Nota-se que $\delta(x, \bar{x}) = 14$. O leitor pode perceber que foi permitida a alteração no modo de execução. Então, percebe-se que, apesar da mudança no modo de execução não influenciar na contagem do grau de diferença, essa mudança permite mais soluções para uma vizinhaça definida por k . Isso permite definir o seguinte em relação à cardinalidade máxima para uma vizinhaça, isto é, o máximo valor que k pode assumir.

Definição 5.2.1. *O limite superior para k é tal que, $k \leq 2n$, sendo n a quantidade de tarefas no projeto.*

O leitor também pode ter percebido que a contagem computada nas expressões (5.2) e (5.3) resultam em um valor par.

Proposição 5.2.1. *Os pseudo-cortes $\delta(x, \bar{x})$ caracterizados pelas expressões (5.2) e (5.3) sempre geram um resultado par.*

Demonstração. Sem perda de generalidade, considere a Expressão (5.2). Sejam $\bar{x}$ e x duas soluções factíveis de P . Assume-se, também, que elas estão no subproblema corrente. Por conveniência, sejam j , m e t , respectivamente, uma tarefa, um modo de execução e o possível tempo de término para a solução x , assim como, sejam $\bar{m}$ e $\bar{t}$, respectivamente, um modo de execução e o possível tempo de término para a solução de referência $\bar{x}$, da mesma tarefa j . Quando j é analisada, os tempos de término são comparados para os respectivos modos de execução. Quando os tempos de término são equivalentes, bem como os modos de execução, isto é, $t = \bar{t}$ e $m = \bar{m}$, então $x_{jmt} = 1$, e portanto, $\sum_{(m,t) \in \bar{S}_j} (1 - x_{jmt}) = 0$. Também, fica claro, que a segunda parte da contagem $\sum_{j=1}^n \sum_{(m,t) \notin \bar{S}_j} x_{jmt} = 0$, uma vez que não há índices distintos entre as duas soluções. Conclui-se que índices iguais para as duas soluções, contribuem com zero (0) para a contagem. Caso contrário, $t \neq \bar{t}$ ou $m \neq \bar{m}$. Quando $t \neq \bar{t}$, fica claro que, independente do modo de execução, $\sum_{j=1}^n \sum_{(m,t) \in \bar{S}_j} (1 - x_{jmt}) = 1$ e, também, $\sum_{j=1}^n \sum_{(m,t) \notin \bar{S}_j} x_{jmt} = 1$. A agregação dessas duas parcelas, no fim, será 2. Quando $m \neq \bar{m}$, então ainda que os tempos de término sejam os mesmos, os modos de execução são distintos e, portanto, $\sum_{j=1}^n \sum_{(m,t) \in \bar{S}_j} (1 - x_{jmt}) = 1$ e, também, $\sum_{j=1}^n \sum_{(m,t) \notin \bar{S}_j} x_{jmt} = 1$. A agregação dessas duas parcelas, no fim, será 2. Conclui-se que índices distintos para as duas soluções, contribuem com 2 para a contagem, para cada tarefa j . $\square$

Ou seja, um índice distinto, m ou t entre duas soluções, contribui em 2 unidades de diferença entre a solução de referência e uma solução vizinha.

5.2.1.3 Caracterização dos Subproblemas pela Literatura

Uma abordagem baseada em B&C para resolução do MRCPSP é apresentada por [Zhu et al. \(2006\)](#). Esses autores definem o conjunto suporte $\bar{S}$, em relação ao pseudo-corte da Expressão (5.3), como sendo:

$$\bar{S}_j = \{t : \max(EFT_j, \bar{t} - \Delta) \leq t \leq \min(LFT_j, \bar{t} + \Delta)\}, j = 1, \dots, n \quad (5.7)$$

para $\bar{t}$ o tempo de término da solução de referência e algum $\Delta \geq 0$. Ou seja, a vizinhança de uma solução de referência $\bar{x}$ é caracterizada pelas soluções que se diferem de $\bar{x}$ em relação a diferentes tempos de término das atividades, porém, dentro da janela temporal $[\max(EFT_j, \bar{t} - \Delta), \min(LFT_j, \bar{t} + \Delta)]$. O conjunto $\bar{S}$ da Expressão (5.7) inclui variáveis que estão a uma distância Δ dos tempos de término das tarefas da solução de referência. Percebe-se que essa definição de [Zhu et al. \(2006\)](#) também reduz o MRCPSP ao RCPSP.

Como a proposta desta tese é investigar a influência da caracterização de uma classe de desigualdades, a Expressão (5.7) foi explorada, em relação ao pseudo-corte da Expressão (5.2) para também definir $\bar{S}$ como:

$$\bar{S}_j = \{(m, t) : \max(EFT_j, \bar{t} - \Delta) \leq t \leq \min(LFT_j, \bar{t} + \Delta)\}, j = 1, \dots, n \quad (5.8)$$

isto é, a vizinhança de uma solução de referência $\bar{x}$ é caracterizada pelas soluções que se diferem de $\bar{x}$ em relação a diferentes modos de execução e tempos de término das atividades, porém, dentro da janela temporal $[\max(EFT_j, \bar{t} - \Delta), \min(LFT_j, \bar{t} + \Delta)]$. Essa definição expande a definição da Expressão (5.7).

Algoritmo 7: SimpleStrongDiversification

Entrada: $\bar{x}, \beta, n$

```

1  $n\_mod = \lceil \beta * n \rceil$  ;
2 codificar  $\bar{x}$  para  $L(\bar{x})$  ;
3 para  $i = 1$  até  $n\_mod$  faça
4    $j =$  escolher alguma tarefa de  $L_J(\bar{x})$  ;
5   trocar a posição de  $j$  em  $L_J(\bar{x})$  ;
6   trocar o modo de  $j$  em  $L_M(\bar{x})$  ;
7 fim
8  $L(\bar{x}_1) = \text{NJM-SHCM}(FILS, L(\bar{x}))$  ;
9  $L(\bar{x}_2) = \text{NJM-SWCM}(FILS, L(\bar{x}_1))$  ;
10 decodificar  $L(\bar{x}_2)$  para  $\bar{x}$  ;
11 return  $\bar{x}$  ;
```

5.2.2 Strong Diversification

Quando o método LB reduz a cardinalidade de uma vizinhança, o valor k pode voltar ao valor k_{min} , evidenciando, então, uma ciclagem, conforme apresentado na linha 30 do Algoritmo 6. Quando isso acontece, há a necessidade de um método de perturbação, destacado em Fischetti e Lodi (2003) como *Strong Diversification*. Esse procedimento auxilia o método LB a escapar de um ótimo local. Nesta tese, são apresentadas três estratégias de diversificação, por meio dos Algoritmos 7, 8 e 9.

Algoritmo 8: StrongDiversification

Entrada: $\bar{x}, \beta, n$

```

1  $n\_mod = \lceil \beta * n \rceil$  ;
2 codificar  $\bar{x}$  para  $L(\bar{x})$  ;
3 para  $i = 1$  até  $n\_mod$  faça
4    $j =$  escolher alguma tarefa de  $L_J(\bar{x})$  ;
5   trocar a posição de  $j$  em  $L_J(\bar{x})$  ;
6   trocar o modo de  $j'$  em  $L_M(\bar{x})$  ;
7 fim
8  $L(\bar{x}_2) = \text{LocalSearch}(L(\bar{x}), n)$  ;
9 decodificar  $L(\bar{x}_2)$  para  $\bar{x}$  ;
10 return  $\bar{x}$  ;
```

Os dois primeiros procedimentos de diversificação descritos nos Algoritmos 7 e 8 modificam uma solução $\bar{x}$ em até $\beta\%$. Essa modificação é a simples troca de posição de uma tarefa, bem como de seu modo de execução. A troca é feita de forma aleatória, desde que seja factível em relação à restrição de precedência e de consumo de recursos não-renováveis. Após essa modificação, a lista $L(\bar{x})$ está alterada. Em seguida, há a busca local sobre essa lista. Por fim, há a decodificação para $\bar{x}$. Esses dois procedimentos diferem na busca local. O Algoritmo 7 usa as próprias estruturas de vizinhança, porém guiadas pela heurística FILS. Já o Algoritmo 8 usa a heurística VND, que será descrita adiante.

O segundo procedimento de diversificação, descrito no Algoritmo 9, cria uma

Algoritmo 9: StrongDiversificationAtRandom

Entrada: n

```

1 criar  $L_J(\bar{x})$  ;
2 criar  $L_M(\bar{x})$  ;
3 reparar  $L_M(\bar{x})$  se não é viável ;
4 se  $L_M(\bar{x})$  é viável então
5   | sequenciar  $L(\bar{x})$  por SSS ;
6   |  $L(\bar{x}_2) = LocalSearch(L(\bar{x}), n)$  ;
7   | decodificar  $L(\bar{x}_2)$  para  $\bar{x}$  ;
8   | return  $\bar{x}$  ;
9 senão
10  | return null ;
11 fim
```

lista $L(\bar{x})$ sem modos de execução ou tempos de término. Em seguida, as tarefas são ordenadas de forma aleatória, bem como seus modos de execução são definidos de forma aleatória. A atribuição aleatória de modos de execução pode ser infactível em relação ao consumo de recursos não-renováveis. Então, um simples procedimento de reparação de infactibilidade é executado sobre a atribuição de modos de execução. Esse procedimento é apresentado por [Hartmann \(2001\)](#), e é descrito conforme segue. Uma tarefa j é selecionada aleatoriamente pela sua posição em $L_J(\bar{x})$. Em seguida, um modo de execução m^1 também é selecionado de forma aleatória, tal que $m^1 \neq \bar{m}$, para $\bar{m}$ representando o modo de execução atual de j . Se m^1 reduz o consumo total de recursos não-renováveis, então m^1 será aceito como o modo de execução para j . Esse processo é repetido até que aconteça n tentativas consecutivas sem sucesso de reduzir o consumo de recursos não-renováveis. Após esse último procedimento, se a atribuição de modos de execução é viável em relação ao consumo de recursos não-renováveis, então as tarefas são sequenciadas e a solução obtida é refinada por um procedimento de busca local.

Os procedimentos usados nesta tese como busca local são baseados nas heurísticas FILS (*First Improvement Local Search*) e BILS (*Best Improvement Local Search*) ([Hoos e Stützle, 2004](#)), bem como na heurística VND (*Variable Neighborhood Descent*) ([Mladenovic e Hansen, 1997](#)), e serão descritos na Seção 5.4.

5.3 Construção de uma Solução Inicial

Para atribuir um conjunto viável de modos de execução para as atividades, é necessário respeitar a restrição de disponibilidade desses recursos. Na formulação (P) , apresentada nas expressões (2.10)-(2.15), considerando apenas a Expressão (2.13), é possível identificar um outro problema de otimização, desconsiderando, é claro, os tempos de término das tarefas. Seja $x_{jm} = 1$ uma tarefa j com modo de execução m ; caso contrário, $x_{jm} = 0$. A formulação (P') a seguir é usada para encontrar uma

atribuição inicial e viável de modos de execução para as tarefas.

$$(P') \quad \min \sum_{j=1}^n d_{jm} \left(\sum_{k=1}^{K^\rho} \frac{r_{jmk}^\rho}{R_k^\rho} \right) x_{jm} \quad (5.9)$$

s.t.

$$\sum_{m=1}^{|M_j|} x_{jm} = 1, j = 1, \dots, n \quad (5.10)$$

$$\sum_{j=1}^n \sum_{m=1}^{|M_j|} r_{jmk}^\nu x_{jm} \leq R_k^\nu, \quad k = 1, \dots, R^\nu \quad (5.11)$$

$$x_{jm} \in \{0, 1\}, \quad j = 1, \dots, n; m = 1, \dots, |M_j| \quad (5.12)$$

A Expressão (5.9) procura pelos modos de execução com o menor consumo relativo dos recursos renováveis por unidade de tempo, isto é, quando uma tarefa consome uma quantidade de recursos significativamente menor que a quantidade disponível, esta relação impacta no tempo de duração na função objetivo. Chamamos a relação $\frac{r_{jmk}^\rho}{R_k^\rho}$ de demanda normalizada de consumo. A justificativa para essa função objetivo é que as tarefas com menores tempo de execução tendem a consumir uma quantidade maior de recursos.

A Expressão (5.10) garante a atribuição de um único modo de execução por tarefa. A Expressão (5.11) garante respeitar a disponibilidade de recursos não-renováveis. Por fim, a Expressão (5.12) apresenta a característica binária da formulação.

A formulação (P') é semelhante ao problema *NP-Hard* conhecido como *Multidimensional 0-1 Knapsack Problem* (Fréville, 2004). Os *solvers* atuais resolvem esse problema na otimalidade, utilizando pouco tempo de execução, como discutido em (Santos et al., 2014).

Após definir os modos de execução, o processo de sequenciamento é utilizar o método de sequenciamento *Serial Scheduling Scheme* (SSS), já descrito.

5.4 Heurísticas de Busca Local

A codificação $L(\bar{x})$ é a usada. Em seguida, serão descritas as estruturas de vizinhança usadas na busca local, e por fim, os algoritmos de busca local.

5.4.1 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 o conjunto de soluções viáveis 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:

- (i) vizinhança baseada em atividades;
- (ii) vizinhança baseada em modos de execução.

As estruturas de vizinhança são adaptadas de [Asta et al. \(2016\)](#) e são baseadas nas heurísticas FILS e BILS:

- (i) NJ-SW: Uma tarefa j é selecionada, aleatoriamente. Então seu predecessor mais próximo p , tal que $(p, j) \in \mathcal{P}$, e seu sucessor mais próximo s , tal que $(j, s) \in \mathcal{S}$, são determinados, respectivamente, nas posições i_p e i_s de L_J . Em seguida, o intervalo $I = [i_p + 1, i_s - 1]$ é determinado para trocar a ordem entre a tarefa j e cada tarefa $\bar{j}$ identificada pela posição $\bar{i} \in I$, tal que $j \neq \bar{j}$. As tarefas j e $\bar{j}$ são trocadas de posição, se e somente se, não há violação da restrição de precedência para $\bar{j}$. Destaque-se, é claro, que se há a troca entre as posições de j e $\bar{j}$ em L_J , há também a troca de posições em L_M . Então, o método SSS novamente realiza o sequenciamento para determinar os novos tempos de término L_T . Aqui, há uma otimização do método SSS para reconstruir o sequenciamento a partir da menor posição determinada pela troca entre j e $\bar{j}$. Esse processo avalia cada intervalo produzido por cada tarefa $j = 1, \dots, n$ (BILS) ou quando o novo *makespan* diminui, então nenhuma outra tarefa j será avaliada (FILS).
- (ii) NJ-SH: Uma tarefa j é selecionada, aleatoriamente, e os valores p e s são determinados para identificar quão longe j pode mover-se para frente ou para trás em L_J , isto é, $I = [i_p + 1, i_s - 1]$ é determinado. Assim, a tarefa j é colocada em cada posição $\bar{i} = i_p + 1, \dots, i_s - 1$, se possível, e o sequenciamento é reconstruído e avaliado. Esse processo avalia cada intervalo produzido por cada tarefa $j = 1, \dots, n$ (BILS) ou quando o novo *makespan* diminui, então nenhuma outra tarefa j será avaliada (FILS).
- (iii) NJM-SWCM: Uma tarefa j é selecionada, aleatoriamente. Em seguida $I = [i_p + 1, i_s - 1]$ é determinado. Para cada posição $\bar{i} \in I$ representando uma tarefa $\bar{j}$, troca-se j e $\bar{j}$ de posição, se possível. Após, cada modo de execução m' de j é avaliado para atribuição, tal que $m' \neq m$, para m representando o modo de execução atual. Um modo de execução é avaliado se, e somente se, é viável em relação ao consumo de recursos não-renováveis. Esse processo avalia para $j = 1, \dots, n$, cada intervalo $I = [i_p + 1, i_s - 1]$ de j , bem como todos os modos de execução de j (BILS), ou quando o novo *makespan* diminui, então nenhuma outra tarefa j será avaliada (FILS).

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1

(a)

L_J	1	3	2	4	6	5	7	8
L_M	1	1	1	1	1	1	1	1

(b)

Figura 5.3: Solução corrente $L(x)$ - em (a) - levando a um vizinho $L(x^1)$ - em (b) - na vizinhança NJ-SH

L_J	1	2	4	6	3	5	7	8
L_M	1	1	1	1	1	1	1	1

(a)

L_J	1	2	4	3	6	5	7	8
L_M	1	1	1	2	1	1	1	1

(b)

Figura 5.4: Solução corrente $L(x)$ levando a um vizinho $L(x^1)$ na vizinhança NJM-SWCM

- (iv) NJM-SHCM: Uma tarefa j é selecionada, aleatoriamente. Em seguida $I = [i_p + 1, i_s - 1]$ é determinado. Para cada posição $\bar{i} \in I$, move-se j para esta nova posição, se possível. Após, cada modo de execução m' de j é avaliado para atribuição, tal que $m' \neq m$, para m representando o modo de execução atual. Um modo de execução é avaliado se, e somente se, é viável em relação ao consumo de recursos não-renováveis. Esse processo avalia para $j = 1 \dots, n$, cada intervalo $I = [i_p + 1, i_s - 1]$ de j , bem como todos os modos de execução de j (BILS), ou quando o novo *makespan* diminui, então nenhuma outra tarefa j será avaliada (FILS).

Como exemplo, as Figuras 5.3 e 5.4 apresentam, respectivamente, uma solução nas respectivas vizinhanças NJ-SH e NJM-SWCM. É importante destacar que essas estruturas de vizinhanças usam o formato em listas $L(x)$. As vizinhanças geradas pelo pseudo-corte são estruturas distintas, que usam a variável binária x_{jmt} .

De acordo com a Figura 5.3, a solução corrente $L(x)$ é modificada escolhendo-se $j = 3$, encontrando seu predecessor e sucessor mais próximos, dados, respectivamente, como $p_3 = 1$ e $s_3 = 5$, e inserindo $j = 3$ em uma nova posição $L(x)[i = 1]$, sem violação da restrição de precedência.

As estruturas de vizinhança que realizam as perturbações devem ser rápidas em suas execuções, proporcionando ao método LB um maior tempo para explorar o espaço de soluções.

5.4.1.1 Descrição da Busca Local

O processo de busca local é realizado pelo método *Variable Neighborhood Descent* (VND) (Mladenovic e Hansen, 1997). A implementação do método VND para o MRCPSP, usando as estruturas de vizinhança descritas anteriormente, é mostrada no Algoritmo 10.

Algoritmo 10: VND adaptado

Entrada: $L(x), n$

```

1  $N = [\text{NJ-SH}, \text{NJ-SW}, \text{NJM-SHCM}, \text{NJM-SWCM}]$  ;
2  $\text{bestmak} = L_T(x)[n]$  ;
3 para  $i = 1$  to  $|N|$  faça
4    $L(x)_1 = N[i](BILS, L(x))$  ;
5   if  $L_T(x)_1[n] < \text{bestmak}$  then
6      $i = 0$  ;
7      $L(x) = \text{copia}(L(x)_1)$  ;
8      $\text{bestmak} = L_T(x)_1[n]$  ;
9   end
10 fim
11 return  $L(x)$  ;
```

A solução viável inicial x é passada como parâmetro para o método VND adaptado, em sua forma de listas, descrito no Algoritmo 10. Em seguida, há a busca local ordenada para cada estrutura de vizinhança, conforme as linhas 3-10. As vizinhanças NJ-SH e NJ-SW, por realizarem uma busca local considerando apenas tarefas, são utilizadas primeiro pelo VND adaptado. Já as vizinhanças NJM-SHCM e NJM-SWCM, que verificam as tarefas e também os modos, são vizinhanças de maior cardinalidade, e, portanto, são verificadas por último. Destaca-se que nessa implementação a heurística BILS foi utilizada.

Capítulo 6

Resultados Computacionais

As variantes do método *Local Branching* propostas nesta tese foram implementadas em linguagem C++, usando o compilador GNU G++ 5.4.0. O *solver* utilizado foi o CPLEX 12.6. Foram realizados testes computacionais em três classes distintas de problemas, i.e., PSPLIB (Kolisch e Sprecher, 1997) e MMLIB/MMLIB+ (Van Petteghem e Vanhoucke, 2014). Além disso, os experimentos computacionais para cada uma destas classes foram executados em computadores distintos, como mostrado a seguir:

- PSPLIB: Intel Core i5-8250U processor; 1.60 GHz; 8GB RAM; and Ubuntu Linux 64-bit operating system.
- MMLIB: Intel Core i7-3770U processor; 3.40 GHz; 32GB RAM; and Ubuntu Linux 64-bit operating system.
- MMLIB+: Intel Xeon E5620 2.40GHz; 112GB RAM; and CentOS Linux 64-bit. Neste computador foram executados todos os problemas MMLIB+.

A justificativa para isso é a disponibilidade das máquinas para a realização dos experimentos em cada um dos referidos conjuntos de instâncias.

As Tabelas 6.1, 6.2 e 6.3 apresentam as características dessas bibliotecas.

Tabela 6.1: PSPLIB classes

Class	J10	J12	J14	J16	J18	J20	J30
# feasible instances	536	547	551	550	552	554	552
# executions modes	3	3	3	3	3	3	3
# renewable resources	2	2	2	2	2	2	2
# non-renewable resources	2	2	2	2	2	2	2

Os resultados computacionais serão apresentados da seguinte forma. A versão original do *Local Branching*, apresentada pelo Algoritmo 5, será utilizada em conjunto com os pseudo-cortes discutidos na Seção 5.2, e o método de perturbação será o Algoritmo 7. Essa variação do LB será identificada como *Local Branching Basic* (LB-B). Após, serão analisados os resultados obtidos pelo Algoritmo 6, que utiliza o método VND como busca local. Essa variante será chamada de *Local Branching Improved* (LB-I).

Tabela 6.2: MMLIB classes

Class	J50	J100
# feasible instances	540	540
# executions modes	3	3
# renewable resources	2	2
# non-renewable resources	2	2

Tabela 6.3: MMLIB+ classes

Class	J50	J100
# feasible instances	1620	1620
# executions modes	{3, 6, 9}	{3, 6, 9}
# renewable resources	{2, 4}	{2, 4}
# non-renewable resources	{2, 4}	{2, 4}

6.1 Resultados para o LB-B

Nesta seção serão apresentados os resultados obtidos pela versão LB-B. Inicialmente, há uma discussão sobre os valores de dois parâmetros desse algoritmo. Em seguida, será discutida a diferença entre as caracterizações de pseudo-cortes, assim como discute-se os outros dois parâmetros. Por fim, serão analisados os resultados da execução do LB-B, bem como sua capacidade de resolução dos problemas PSPLIB. Para os problemas MMLIB, os resultados serão apresentados na próxima seção.

6.1.1 Tuning dos parâmetros

A calibração dos parâmetros tem grande importância e influência nos resultados obtidos por procedimentos computacionais. Nesta tese, o pacote IRACE (López-Ibáñez et al., 2016) foi usado para calibrar os parâmetros da metaheurística LB-B. O pacote IRACE automatiza todo o planejamento estatístico para apresentar amostras, estatisticamente distintas. Essas amostras são definidas pelas soluções obtidas pelo método LB-B com diferentes valores para os parâmetros. Esse experimento foi conduzido da seguinte maneira.

Foram selecionadas, aleatoriamente, 200 instâncias da classe J20 PSPLIB. Essas instâncias são as de maior dimensão com valores de ótimos conhecidos. Cada instância selecionada foi solucionada uma vez, e, então, foi coletado o desvio médio para o ótimo conhecido. O critério de parada foi definido para 1800 segundos, com tempo limite por subproblema igual a 90 segundos, usando a caracterização da Expressão (5.5). Entre as instâncias selecionadas, 100 delas possuem o valor *Resource Factor* (RF) dado por $RF = 0,5$; as demais 100 instâncias possuem o valor $RF = 1$. Esse fator indica a quantidade demandada, em média, de cada recurso, para cada combinação tarefa-modo. Quando $RF = 1$, cada par tarefa-modo consome pelo menos uma unidade de cada recurso; por outro lado, quando $RF < 1$, há pares de tarefa-modo que não consomem qualquer recurso. De acordo com a literatura, esse fator indica o quanto é difícil resolver a instância.

Tabela 6.4: Primeira combinação de valores para os parâmetros.

Parâmetros	Níveis
k_0	(10, 15, 20, 25 , 30)
β	(20, 30, 40, 50)

Tabela 6.5: Segunda combinação de valores para o parâmetro de interesse.

Parâmetro	Níveis
β	(50, 80 , 100)

Dois conjuntos de valores para os parâmetros são usados para analisar a influência desses parâmetros. O primeiro conjunto é apresentado na Tabela 6.4. Após fixar o parâmetro k_0 , o segundo conjunto de valores é apresentado na Tabela 6.5. Os valores destacados em negrito apresentaram os melhores resultados. Assim, esses valores serão usados para os próximos testes computacionais discutidos para a variante LB-B.

6.1.2 Caracterizações do pseudo-corte

Nesse experimento será verificada a influência da caracterização do pseudo-corte no método LB. As Expressões (5.5), (5.6), (5.7) e (5.8) são usadas para análise do resultado. O critério de parada foi definido para 1800 segundos, com tempo limite por subproblema igual a 90 segundos. Para esse experimento, toda a classe de problemas PSPLIB foi resolvida. As Tabelas 6.6 e 6.7 apresentam os resultados. Nessas tabelas, $\#opt$ representa o número de ótimos encontrados, $\%opt$ representa o percentual de ótimos encontrados, $\%gap_{opt}$ representa o desvio médio para o ótimo, e o valor $\%gap_{cpm}$ significa o desvio médio para o caminho crítico.

Em relação aos resultados da Tabela 6.6, percebe-se que, ao usar a caracterização da Expressão (5.6), foram obtidos os melhores resultados, isto é, possui melhor capacidade para ajudar o método LB a explorar o espaço de soluções. Essa afirmação é clara ao se analisar as classes J16, J18, J20 e J30. Como a Expressão (5.6) permite um subproblema de maior cardinalidade, o método LB pode avaliar um maior número de soluções em cada subproblema durante a busca. Como destaque, é importante mencionar os resultados para a classe J30 para o número de ótimos encontrados, em relação à Expressão (5.6), que foi 10% maior que a outra caracterização, além de 33% a menos no valor do desvio médio.

Para a Tabela 6.7, a Expressão (5.7), apresentou, em média, os melhores resultados. O interessante é que essa caracterização, proposta por [Zhu et al. \(2006\)](#), gera subproblemas de menor cardinalidade que a Expressão (5.8). Percebe-se que os resultados dessa última tabela não são melhores dos que os resultados da Tabela 6.6 ao se analisar a Expressão (5.6).

Conclui-se, então, de acordo com esses resultados, que a caracterização de um subproblema pela Expressão (5.6), proposta nesta tese, apresenta os melhores resultados para que o método LB explore o espaço de soluções. Essa é a primeira

Tabela 6.6: Resultados PSPLIB (J10-J30) para as Expressões (5.5 e 5.6).

	$\bar{S}$ definido pela Expressão (5.5)				$\bar{S}$ definido pela Expressão (5.6)			
	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$
J10	536	100	0	–	536	100	0	–
J12	547	100	0	–	547	100	0	–
J14	551	100	0	–	551	100	0	–
J16	544	98.9091	0.1112	–	550	100	0	–
J18	528	95.6522	0.1955	–	547	99.0	0.040	–
J20	518	93.5018	0.2839	–	530	96.0433	0.1351	–
J30	438	74.7246	1.8591	15.6529	453	82.5362	1.2383	13.1771

Tabela 6.7: Resultados PSPLIB (J10-J30) para as Expressões (5.7 e 5.8).

	$\bar{S}$ definido pela Expressão (5.7)				$\bar{S}$ definido pela Expressão (5.8)			
	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$
J10	536	100	0	–	536	100	0	–
J12	547	100	0	–	547	100	0	–
J14	551	100	0	–	551	100	0	–
J16	544	98.9391	0.0921	–	545	99.0909	0.0885	–
J18	536	97.1014	0.1027	–	533	96.5580	0.1197	–
J20	521	94.0433	0.2353	–	517	93.3213	0.2902	–
J30	447	80.9783	1.2295	14.0899	433	78.4420	1.5389	14.5256

contribuição desta tese, ao propor uma nova caracterização de um pseudo-corte, que apresenta melhores resultados do que aqueles já conhecidos pela literatura, para o método de busca, porém com uma outra caracterização de pseudo-corte.

Para uma melhor análise dos resultados obtidos, os problemas J20 e J30 foram separados pelo valor RF. A Tabela 6.8 apresenta os resultados somente para as caracterizações pelas Expressões (5.6) e (5.7). É possível perceber que, quanto maior a demanda de recursos, mais difícil foi para resolver a instância. Os resultados reforçam a qualidade do pseudo-corte caracterizado pela Expressão (5.6), em relação àquela caracterização proposta por Zhu et al. (2006).

Para uma melhor compreensão dos resultados obtidos pela caracterização de pseudo-corte da Expressão (5.6) para o método LB, outro teste computacional foi realizado, porém desta vez variando os valores para os parâmetros `stop_criterion` (nomeado, por conveniência, para `stc`) e `node_time_limit` (nomeado para `ntl`), para solucionar as classes J20 and J30. Por meio dessa variação de valores para esses parâmetros, é possível perceber como o método LB se comporta em relação ao tempo total de execução (`tot_rt`), o tempo de execução da fase de perturbação (`ls_rt`), o número de nós explorados pelo solver CPLEX (`avg_nodes`), e o número de pseudo-cortes, em média, adicionados (`avg_psc`). Então, após essa análise, também será possível entender sobre a tendência desses parâmetros. A Tabela 6.9 resume os resultados.

A partir dos resultados da Tabela 6.9, percebe-se que, quando o método LB possui mais tempo para explorar os subproblemas, e também, maior tempo total de

Tabela 6.8: Resultados para PSPLIB (J20 e J30) pelas caracterizações das Expressões (5.6) e (5.7)

		J20		J30	
		$RF = 0.5$	$RF = 1$	$RF = 0.5$	$RF = 1$
Expression (5.6)	%gap	0.000000	0.277197	0.288364	2.009556
Expression (5.7)	%gap	0.023391	0.441729	0.336282	2.088222

Tabela 6.9: Análise no tempo de execução para PSPLIB (J20-J30) pela caracterização da Expressão (5.6).

J20							
(ntl, stc)	opt (%)	gap_{opt} (%)	gap_{cpm} (%)	tot_rt (sec.)	ls_rt (sec.)	avg_nodes	avg_psc
(90, 1800)	96.0433	0.1351	—	168.846	0.0001	16409	6
(120, 3600)	97.2094	0.1182	—	242.142	0.0001	22652	7
(120, 5400)	98.4534	0.0818	—	252.276	0.0002	25930	7
(180, 7200)	98.9534	0.0630	—	320.34	0.0001	36621	7

J30							
(ntl, stc)	$best$ (%)	gap_{best} (%)	gap_{cpm} (%)	tot_rt (sec.)	ls_rt (sec.)	avg_nodes	avg_psc
(90, 1800)	82.5362	1.2383	13.1771	607.392	0.006	37639	15
(120, 3600)	84.0797	1.1924	12.2970	1174.254	0.006	53465	21
(120, 5400)	86.8913	1.0741	11.1401	1669.896	0.012	77251	28
(180, 7200)	89.4275	0.8793	10.7324	1912.572	0.007	95047	23

execução, ele encontra melhores resultados, o que é um comportamento esperado. Outra percepção é sobre o tempo de execução, em média, da fase de perturbação, que mostra-se muito pequeno. Isso corrobora a afirmação feita sobre esse procedimento ser rápido o suficiente para deixar a fase de exploração dos subproblemas com maior tempo. Porém, isso também pode ser um indício de que um método de perturbação, aliado a uma busca local mais robusta, apesar de consumir mais tempo, pode entregar melhores soluções de referência. Essa característica será verificada mais a frente, neste Capítulo. Uma outra análise desse resultado é o valor (`tot_rt`), que, em média, não excede 25% do tempo limite de execução, para a classe J20, e não excede 40% desse mesmo tempo para a classe J30. Então, é possível concluir que o método LB, por meio da caracterização dada pela Expressão (5.6), tende a encontrar soluções suficientemente boas, quando não encontra as soluções ótimas, antes do tempo limite para a execução desse método. A Figura 6.1 apresenta esses resultados por meio de *boxplots*.

Por meio das informações contidas na Figura 6.1, fica claro que os parâmetros `tot_rt` e `avg_nodes` são aqueles que apresentam a maior variação, quando há um aumento no tempo de execução para o LB-B, para a classe J30. Por outro lado, os parâmetros $\%gap_{opt}$ e $\%gap_{best}$ não apresentaram essa variação significativa, exceto pelo *outliers*. Quando se analisa a classe J20, o método LB-B se apresenta estável, em relação à grande maioria dos problemas resolvidos, de acordo com os quartis, embora não será afirmado que todas as amostras analisadas não são estatisticamente

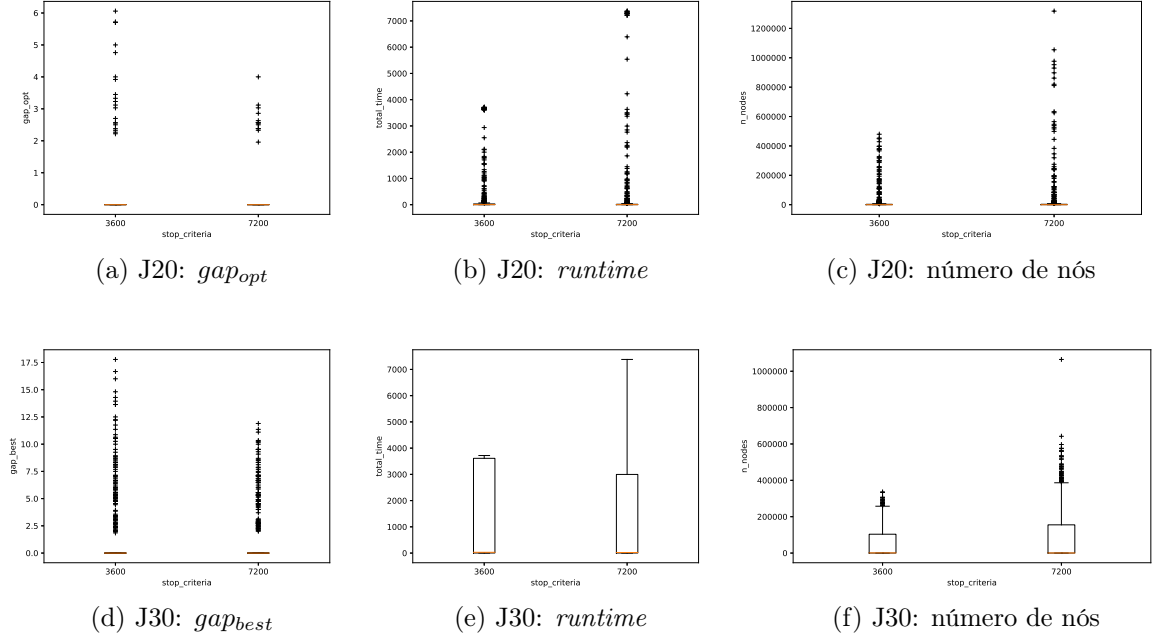


Figura 6.1: Análise no tempo de execução, bem como gap_{opt} , gap_{best} e número de nós, ao resolver os problemas PSPLIB J20 e J30, pelo método LB-B Expressão (5.6), para critério de parada com 3600 e 7200 segundos.

distintas, devido ao número de *outliers*.

6.2 Resultados para o LB-I

Nesta seção serão apresentados os resultados obtidos pela variante LB-I descrita no Algoritmo 6, que representa uma versão aprimorada do Algoritmo LB-B apresentado na Seção 6.1. Serão discutidos os efeitos dos parâmetros, e em seguida, são apresentados os resultados obtidos, assim como é discutida uma comparação com os resultados obtidos por métodos no estado-da-arte presentes na literatura para resolução do MRCPSP.

6.2.1 Tuning dos parâmetros

Novamente, foram solucionadas 200 instâncias J20 (PSPLIB) para verificar a influências dos parâmetros. As mesmas instâncias usadas nos testes computacionais descritos na Seção 6.1 foram usadas nos testes computacionais a seguir. O pseudo-corte utilizado foi aquele caracterizado na Expressão (5.6), pois foi a caracterização que apresentou os melhores resultados. O critério de parada para o método LB-I foi configurado para 1800 segundos, embora serão apresentados resultados com outro valor para o critério de parada.

Dois conjuntos de valores foram analisados para validar a influência dos parâmetros `node_time_limit`, k_0 , and β . O primeiro conjunto é apresentado na Tabela 6.10. Esses valores foram escolhidos de forma arbitrária. Após manter o

Tabela 6.10: Primeira combinação de valores para os parâmetros.

Parameters	Levels
<code>node_time_limit</code>	(90, 120, 180)
k_0	(2, 6 , 10, 20)
β	(20, 40, 90, 80)

Tabela 6.11: Segunda combinação de valores para os parâmetros.

Parameters	Levels
<code>node_time_limit</code>	(180 , 200, 220, 240)
β	(80 , 100)

parâmetro $k_0 = 6$, o segundo conjunto de valores é analisado e apresentado na Tabela 6.11. Esses últimos valores foram escolhidos ao analisar a tendência de valores no primeiro conjunto. As Tabelas 6.10 e 6.11 destacam em negrito os resultados de melhor desempenho.

Nessa variante do LB, os parâmetros k_0 e `node_time_limit` foram aqueles que apresentaram uma mudança significativa, em relação aos valores das Tabelas 6.4 e 6.5. O que se pode afirmar sobre o parâmetro k começando com um valor menor é que, possivelmente, mais pseudo-cortes serão inseridos na formulação do MRCPSP, pelo menos para instâncias suficientemente pequenas. Em relação a um tempo maior para resolver um subproblema, em relação à versão do LB-B mostrada na Seção 6.1, pode-se comentar que na variante LB-I foi melhor o método explorar um subproblema por mais tempo. Essa análise pode ser justificada por, somente, as seguintes condições: (i) a otimização no pseudo-corte, evitando uma parcela de um subproblema ser verificada novamente, de acordo com os parâmetros k_{min} e k_{aux} ; e, (ii) pela mudança no processo de perturbação, que encontra uma solução de referência de melhor qualidade.

6.2.2 Resultados em relação à Literatura

Os resultados durante o processo de *setup* dos parâmetros já se mostraram melhores do que aqueles alcançados pela variante LB-B discutida anteriormente. Para apresentar essa diferença, esta seção discute os resultados separados segundo as classes PSPLIB e MMLIB, incluindo os resultados de maior relevância na literatura para resolução do MRCPSP.

6.2.2.1 Resultados para PSPLIB

Para validar os resultados alcançados pelo método LB, foi realizada uma comparação com os trabalhos de Zhu et al. (2006) e Schnell e Hartl (2017), uma vez que apresentam abordagens via matheurísticas. Esses trabalhos reportam resultados significativos ao aplicar métodos de programação matemática para resolução do MRCPSP. O trabalho de Schnell e Hartl (2017) apresenta resultados para 6 abordagens distintas baseadas em Programação por Restrições, com algumas dessas

Tabela 6.12: Comparação entre os resultados para os problemas J20 e J30 da PSPLIB, por [Zhu et al. \(2006\)](#) via LB, G12Max e SCIPMax de [Schnell e Hartl \(2017\)](#), a LB-B e LB-I. O símbolo “–” se refere a valores não publicados.

Método	J20		J30		
	#opt	%gap _{opt}	#best	%gap _{best}	%gap _{cpm}
LB: (Zhu et al., 2006)	554	0	529	–	–
G12Max ¹	552 ²	0.05 ²	521 ³	2.59 ³	12.93 ³
SCIPMax ¹	547 ²	0.32 ²	508 ³	3.37 ³	13.32 ³
LB-B ⁴	532	0.1351	455	1.2383	13.1771
LB-B ⁵	548	0.0630	494	0.8793	10.7324
LB-I ⁴	539	0.0990	462	1.0593	13.0202
LB-I ⁵	551	0.0474	502	0.7037	10.5148

¹ por ([Schnell e Hartl, 2017](#)).

² critério de parada: 1200 segundos.

³ critério de parada: 2400 segundos.

⁴ critério de parada: 1800 segundos.

⁵ critério de parada: 7200 segundos.

abordagens executadas com processamento paralelo. Foram usados para comparação os resultados deste último trabalho, usando o caso de processamento serial. A configuração de computador usada foi Intel Westmere X5650, 2.66 GHz. [Zhu et al. \(2006\)](#) utilizou uma abordagem baseada no método *Local Branching*, em um computador Intel Xeon, 1.80 GHz, 24 GB RAM.

A Tabela 6.12 apresenta os resultados comparativos. Nessa tabela, #opt refere-se ao número de ótimos encontrados; #best refere-se ao número de melhores soluções conhecidas encontradas; gap_{opt} representa o desvio para o ótimo, e gap_{cpm} representa o desvio médio para o caminho crítico.

A versão da matheurística LB-I apresentou resultados melhores do que os obtidos pela versão LB-B, sendo ambas executadas em um computador Intel Core i5-8250U processor; 1.60 GHz; 8GB RAM. Em comparação aos resultados de [Zhu et al. \(2006\)](#), percebe-se que o método LB-I aproximou-se dos resultados de [Zhu et al. \(2006\)](#) para a classe J20, e ficando abaixo para a classe J30, embora [Zhu et al. \(2006\)](#) não apresenta os resultados para o gap_{cpm}, que informa o desvio médio para o caminho crítico. Esse resultado é o que se destaca para a versão LB-I. Reitera-se, também, que em [Zhu et al. \(2006\)](#) não é apresentada uma descrição sobre o algoritmo genético usado para realizar a fase de diversificação, isto é, o algoritmo genético é apresentado como uma *black-box*. Compreende-se que o algoritmo genético possui grande influência no resultado e que o método LB em [Zhu et al. \(2006\)](#) foi responsável por certificar a otimalidade dos resultado obtidos pelo algoritmo genético. Por fim, em comparação com [Schnell e Hartl \(2017\)](#) vs LB-I, os resultados obtidos são semelhantes para a classe J20. Para a classe J30, [Schnell e Hartl \(2017\)](#) foi capaz de encontrar maior quantidade das melhores soluções conhecidas, entretanto, a matheurística LB-I, apresentou em geral melhores soluções em relação ao limite inferior. Isso significa que, apesar de não encontrar a maior quantidade de ótimos/melhores soluções conhecidas para as classes J20/J30, o método LB-I encontra,

em média, soluções de boa qualidade.

6.2.2.2 Resultados para a classe MMLIB

Os resultados obtidos após a resolução da classe MMLIB são apresentados em comparação com os melhores resultados para abordagens exatas e heurísticas. Os métodos LB-B e LB-I foram executados para a solução dessa classe de instâncias em um computador Intel Core i7-3770U processor; 3.40 GHz; 32GB RAM.

Tabela 6.13: Comparação entre os resultados para os problemas J50 e J100 da MMLIB, por [Schnell e Hartl \(2017\)](#), LB-B e LB-I.

	J50 (540 instances)			J100 (540 instances)		
	$\#opt$	$\%gap_{cpm}$	tot_rt (sec.)	$\#opt$	$\%gap_{cpm}$	tot_rt (sec.)
SCIPMax ²	405	34.21	1409.69	312	154.19	3127.93
G12Max ²	363	27.61	1952.36	150	25.71	5963.62
LB-B ¹	211	25.61	2205.11	134	32.41	2876.80
LB-I ¹	216	23.66	2032.13	136	30.46	2632.95
LB-I ³	227	21.79	3733.34	162	23.25	4250.28

¹ critério de parada: 3600 segundos.

² critério de parada: 5400 segundos, para J50, e 7200 para J100.

³ critério de parada: 7200 segundos.

Em relação às abordagens exatas, a Tabela 6.13 apresenta os resultados obtidos pelos métodos implementados nesta tese e os compara com aqueles obtidos em [Schnell e Hartl \(2017\)](#). Nota-se que a abordagem SCIPMax proposta por [Schnell e Hartl \(2017\)](#) obteve melhores quantidades de ótimos. Também percebe-se que a variante LB-I obteve melhores resultados que a abordagem LB-B, usando a mesma configuração de computador, o que permite afirmar que as diferenças entre elas melhoram o desempenho do *Local Branching*, isto é, a importância de uma busca local mais sofisticada. Por fim, percebe-se que, para a classe J50, a abordagem LB-I encontra, em média, melhores valores de $\%gap_{cpm}$ para a classe J50, e para a classe J100, dentre todos os resultados apresentados, isto é, quando não encontrou a solução ótima, essa abordagem encontrou uma solução de boa qualidade, superando, em média, aquelas encontradas em [Schnell e Hartl \(2017\)](#).

Comparando agora os resultados em relação às abordagens baseadas em metaheurísticas, apresentam-se os resultados de [Van Peteghem e Vanhoucke \(2014\)](#), [Geiger \(2017\)](#), [Muritiba et al. \(2018\)](#) e [Chakraborty et al. \(2020\)](#). A Tabela 6.14 mostra os resultados. Nessa tabela, o valor gap_{cpm} indica a quantidade de ótimos obtidos que possuem valor igual ao caminho crítico. O valor gap_{cpm} indica o desvio médio para o caminho crítico. Ao realizar essa comparação, pode-se perceber ainda mais o potencial da matheurística LB adaptada, na resolução de instância de grande porte. Os trabalhos de [Van Peteghem e Vanhoucke \(2014\)](#), [Geiger \(2017\)](#), [Muritiba et al. \(2018\)](#) e [Chakraborty et al. \(2020\)](#) adotam como critério de parada o número de sequenciamentos criados. [Van Peteghem e Vanhoucke \(2014\)](#) afirma que esse critério de parada (i) não é recomendado para todas as diferentes estratégias baseadas

Tabela 6.14: Comparação entre os resultados para os problemas J50 e J100 da MMLIB, entre abordagens no estado-da-arte via metaheurísticas e a variante LB-I. Resultados da literatura referem-se a 5000 e 50000 sequenciamentos gerados. O símbolo “—” refere-se a valores não publicados.

Trabalho/Método	#opt	J50		#opt	J100	
		% gap _{cpm} 5000	% gap _{cpm} 50000		% gap _{cpm} 5000	% gap _{cpm} 50000
(Van Peteghem e Vanhoucke, 2014)	212 ³	25.45	23.79	236 ³	26.51	24.02
(Geiger, 2017)	216 ³	33.02	28.35	236 ³	44.11	32.91
(Muritiba et al., 2018)	—	24.24	23.52	—	25.18	24.01
(Chakraborty et al., 2020)	220 ³	35.6	31.92	236 ³	65.7	60.24
(Fernandes e de Souza, 2021) ¹	211	25.61		134	32.41	
LB-I ¹	216	23.66		136	30.46	
LB-I ²	227	21.79		162	23.25	

¹ critério de parada: 3600 segundos.

² critério de parada: 7200 segundos.

³ Número de ótimos *apud* Chakraborty et al. (2020). Esses valores em Curitiba et al. (2018) não são publicados.

em heurísticas, e (ii) que essas abordagens consomem tempos distintos para construir um sequenciamento. O objetivo com os resultados apresentados nessa tabela é, como já informado, mostrar o potencial do método LB, pois não é indicada uma comparação com critérios distintos de parada.

Para a classe J50, usando 3600 segundos como critério de parada, a matheurística LB-I possui bom desempenho em relação ao gap_{cpm} , assim como apresentou bom desempenho executando em 7200 segundos, ao analisar a mesma variável de interesse, bem como encontrou uma quantidade significativa de soluções ótimas. Para a classe J100, é possível concluir essa mesma análise, destacando a matheurística LB-I ao verificar os valores de gap_{cpm} . É importante destacar que os tempos de execução de uma matheurística são maiores em relação aos tempos de execução de procedimentos heurísticos. Portanto, reforça-se o objetivo de apenas mostrar o potencial do método LB, e não afirmar que esse método superou as abordagens via heurística.

6.2.2.3 Resultados para a classe MMLIB+

Nesta seção, serão apresentados os resultados obtidos somente pela variante LB-I, que foi executada em um computador Intel Xeon E5620 2.40GHz; 112GB RAM.

As Tabelas 6.15 e 6.16 apresentam os resultados detalhados por cada classe de problemas MMLIB+. O critério de parada adotado foi de 7200 segundos. Nessas tabelas, os valores #opt, gap_{cpm}(%), tot_rt, tot_l_rt e avg_nodes, representam, respectivamente, a quantidade de soluções ótimas encontradas, o desvio médio para o caminho crítico, o tempo médio de execução, o tempo médio de execução somente da busca local e o número de nós explorados durante a busca.

Em relação aos problemas J50-MMLIB+, cujos resultados são apresentados na Tabela 6.15, pode-se perceber um padrão nos resultados ao analisar os problemas agrupados por modos de execução. Para 3 modos de execução, houve um aumento

Tabela 6.15: Resultados para cada classe de problemas J50 MMLIB+. O formato apresentado da classe é X-Y-Z, sendo X a quantidade de modos de execução, Y a quantidade de recursos renováveis e Z a quantidade de recursos não-renováveis.

class	#opt	gap _{cpm} (%)	tot_rt (sec.)	tot_l_rt (sec.)	avg_nodes
3-2-2	33	30.14	5152.79	1.95	43508
3-2-4	36	31.57	5178.55	1.82	46515
3-4-2	17	42.47	6775.24	39.96	850
3-4-4	10	50.1	6864.2	42.61	886
6-2-2	6	109.47	7122.02	10.99	822
6-2-4	7	112.94	7194.61	7.79	821
6-4-2	6	120.79	7271	90.93	109
6-4-4	6	123.78	7253.91	97.39	14
9-2-2	5	190.34	7280.28	78.51	0
9-2-4	5	197.26	7279.05	70.73	0
9-4-2	5	199.06	7287.25	131.45	0
9-4-4	4	210.11	7277.24	124.54	0
total	140				
avg		118.2	6828	58.2	7794

Tabela 6.16: Resultados para cada classe de problemas J100 MMLIB+. O formato apresentado da classe é X-Y-Z, sendo X a quantidade de modos de execução, Y a quantidade de recursos renováveis e Z a quantidade de recursos não-renováveis.

class	#opt	gap _{cpm} (%)	tot_rt (sec.)	tot_l_rt (sec.)	avg_nodes
3-2-2	24	41.69	6150.55	19.16	3644
3-2-4	14	49	6912.53	38.49	216
3-4-2	8	48.59	7198.14	517.34	0
3-4-4	9	49.18	7191.02	688.11	0
6-2-2	6	124.09	7294.2	602.82	0
6-2-4	6	131.31	7290.22	581.55	0
6-4-2	6	126.49	7314.44	973.79	0
6-4-4	5	137.03	7309.83	870.33	0
9-2-2	4	206.14	7319.35	833.59	0
9-2-4	4	219.56	7323.71	767.15	0
9-4-2	3	197.74	7332.98	1090.16	0
9-4-4	1	218.13	7344.18	1068.24	0
total	90				
avg		129.1	7165.1	670.9	321.8

significativo no gap_{cpm} , bem como no tempo total de execução, quando aumentaram os recursos renováveis; isso proporcionou uma maior dificuldade na resolução do problema, justificada pelo menor número de nós explorados na busca pelo método LB-I

Tabela 6.17: Comparação entre os resultados para os problemas J50 e J100 da MMLIB+, entre abordagens no estado-da-arte via metaheurísticas e a variante LB-I. Resultados da literatura referem-se a 5000 e 50000 sequenciamentos gerados. O símbolo “—” refere-se a valores não publicados.

Trabalho/Método	#opt	% gap _{cpm}	
		5000	50000
Geiger (2017)	—	149.5	114.22
Muritiba et al. (2018)	—	86.84	83.66
Chakraborty et al. (2020)	—	48.37	36.1
LB-I ¹	230	123.7	

¹ critério de parada: 7200 segundos.

e, um maior esforço computacional para a busca local. Para 6 e 9 modos de execução, essa mesma análise é verdadeira, exceto pelo aumento significativo do gap_{cpm} . Percebe-se ainda que, para os problemas com 9 modos de execução, os resultados obtidos são exclusivos da busca local, uma vez que nenhum nó foi completamente explorado, isto é, a primeira vizinhança não foi resolvida para que novos nós fossem explorados. Fica evidente, também, que a busca local implementada pelo método VND executa em um tempo muito pequeno comparado ao tempo, em média, de execução do método LB-I.

A mesma análise anterior é satisfeita para os problemas J100-MMLIB+, destacando ainda que a busca local ficou responsável por realizar a busca pelo espaço de soluções a partir de 3 modos de execução. Esse resultado é importante, pois se percebe que, ao implementar uma busca local que demanda mais tempo de execução, é possível melhorar ainda mais os resultados e, para problemas em larga escala, o método LB-I teria a responsabilidade de certificar a otimalidade de uma solução obtida pela busca local mais sofisticada, como por exemplo, pela metaheurística VNS (*Variable Neighborhood Search*).

A Tabela 6.17 apresenta os resultados discutidos anteriormente e comparados com a literatura para propostas via metaheurísticas. Novamente, o objetivo é apresentar o potencial alcançado pela metaheurística LB-I, uma vez que os critérios de parada são distintos.

Nota-se, pela Tabela 6.17, que o método LB-I proposto nesta tese não obtém melhores resultados de gap do que os outros trabalhos, embora reportam-se 140 soluções ótimas encontradas.

É importante destacar que até o momento da escrita desta tese, não há na literatura, pelo menos segundo a pesquisa bibliográfica realizada, resultados para a classe MMLIB+ envolvendo metaheurísticas, sendo então, o método LB adaptado a primeira metaheurística proposta a solucionar a classe MMLIB+.

Capítulo 7

Considerações Finais

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. Nesta tese 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). Para a resolução desse problema foi apresentado um esquema de matheurística baseado no método *Local Branching* (LB), sendo que a proposta é apresentar uma classe de desigualdades para a estratégia LB, além de definir uma nova variante desse método, investigando, também, o efeito dos parâmetros. As classes de instâncias das bibliotecas públicas PSPLIB e MMLIB foram solucionadas. Inicialmente, foi verificada a influência dos parâmetros. Em seguida, foi verificada a influência da caracterização de estruturas de vizinhanças que definem um subproblema durante a enumeração da matheurística LB. Por fim, os resultados são comparados com o estado-da-arte.

Duas propostas para o método LB são apresentadas, sendo que, dentre as principais diferenças, listam-se a fase de construção da solução inicial, o método de busca local, a fase de perturbação e o parâmetro que determina o tamanho de um subproblema a ser explorado. Após analisar os resultados, essas diferenças tiveram impacto na qualidade dos resultados objetivos entre as versões implementadas.

Ao comparar os melhores resultados obtidos pelo método LB adaptado nesta tese, em relação aos obtidos pela literatura, foi verificado que restaram apenas 3 instâncias para que se resolvesse completamente a classe J20 da biblioteca PSPLIB. Para essa mesma biblioteca de instâncias, porém, para a classe J30, foram obtidos os melhores valores de **gap** em relação aos resultados de [Zhu et al. \(2006\)](#) e [Schnell e Hartl \(2017\)](#), considerados o estado-da-arte. Em relação à biblioteca MMLIB, também foram obtidos os melhores valores de **gap** para a classe J50 e J100 com um tempo médio de execução menor comparado com os resultados de [Schnell e Hartl \(2017\)](#), considerando apenas soluções via matheurísticas. Quando os resultados foram comparados com as metaheurísticas no estado-da-arte, o método LB adaptado nesta tese, chamado de LB-I, obteve os melhores valores de **gap** para as classes J50 e J100, entretanto destacando a diferença no tempo de execução que está a favor das técnicas de metaheurísticas. Esses resultados destacam uma boa performance do método LB adaptado nesta tese.

Analisando os resultados para resolução da classe MMLIB+, constatou-se que,

para os problemas com maiores modos de execução e com maior quantidade de recursos renováveis, o maior esforço computacional ficou para o método de perturbação que adapta a heurística VND. Este resultado é importante, pois conclui-se que uma técnica de perturbação mais robusta poderá proporcionar melhores soluções para que o método LB utilize essas boas soluções e assim, excluir mais rapidamente partes do espaço de solução. Ao comparar os resultados com aqueles de destaque na literatura, verifica-se que o método LB-I não superou os melhores resultados de todos os trabalhos no estado-da-arte, entretanto destaca-se que a técnica de *Local Branching*, proposta como uma variante nesta tese, é a primeira técnica via matheurística apresentada para resolução da classe MMLIB+ e, que de fato, foi competitiva em relação a alguns dos trabalhos no estado-da-arte.

Conclui-se, então, que a matheurística implementada nesta tese obteve melhores resultados em alguns problemas das bibliotecas públicas analisadas, quando comparada com o estado-da-arte em relação a métodos híbridos entre técnicas exatas e heurísticas para resolução do MRCPSP, justificando que os objetivos desta pesquisa foram alcançados, proporcionando o desenvolvimento de uma variante competitiva de um método conhecido na literatura, porém pouco explorado para a resolução do MRCPSP.

7.1 Propostas para Trabalhos Futuros

Como propostas de continuidade dessa pesquisa, percebe-se que outras técnicas de busca local, diferentes do VND, como por exemplo, VNS ou ILS, poderiam ser usadas, uma vez que possuem melhor características de busca global. Essa possibilidade é justificada pelo tempo total de execução analisado nesta tese, uma vez que ficou evidente que metaheurísticas com melhores mecanismos de busca global tendem a executar por mais tempo. Apesar dessa característica retirar tempo de execução do método LB, o autor desta tese destaca que essas abordagens serão fundamentais para se obter melhores resultados. Uma outra possibilidade de continuidade é fazer o estudo de cortes válidos para o MRCPSP, transformando o método LB em um método B&C, devido ao uso de desigualdades válidas. Também podem ser exploradas outras formulações do MRCPSP, verificando o impacto tanto de pseudo-cortes sobre essas formulações, quanto também o impacto de cortes válidos.

7.2 Publicações

Esta trabalho, ao longo de seu desenvolvimento, iniciado quando do mestrado e tendo sua continuidade no trabalho, deu origem a uma série de artigos acadêmicos, listados a seguir, em sua ordem histórica:

- *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 múlti-*

plos 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;
- *Algoritmo de Seleção Clonal Baseado em Regras de Prioridade para o Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução*. XLVIII Simpósio Brasileiro de Pesquisa Operacional, SBPO 2016;
- *A clonal selection algorithm with local search for solving the multimode resource-constrained project scheduling problem*. Workshop on Applied Combinatorial Optimization Methods, WACOM 2016.
- *A CLONALG-inspired algorithm with adaptive large neighborhood search for the multi-mode resource-constrained project scheduling problem*. In: Metaheuristics International Conference (MIC 2017), 2017, Barcelona. Metaheuristics International Conference (MIC 2017), 2017.
- *Uma Estratégia Local Branching para o Problema de Sequenciamento em Projetos com Restrição em Recursos e Múltiplos Modos de Execução*. In: Simpósio Brasileiro de Pesquisa Operacional (SBPO 2018), 2018, Rio de Janeiro. 50^o Simpósio Brasileiro de Pesquisa Operacional, 2018.
- *A Matheuristic Approach to the Multi-mode Resource Constrained Project Scheduling Problem*. Computers & Industrial Engineering, v. 160, p. 107592, 2021.

Referências Bibliográficas

Afshar, Mohammad Reza; Shahhosseini, Vahid e Sebt, Mohammad Hassan. (2022). A genetic algorithm with a new local search method for solving the multimode resource-constrained project scheduling problem. *International Journal of Construction Management*, v. 22, n. 3, p. 357–365.

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, 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.

Almeida, Bernardo F.; Correia, Isabel e Saldanha-da-Gama, Francisco. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. *ITOR*, v. 26, n. 3, p. 946–967. doi: 10.1111/itor.12568.

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.

Asta, Shahriar; Karapetyan, Daniel; Kheiri, Ahmed; Özcan, Ender e Parkes, Andrew J. (2016). Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences*, v. 373, p. 476–498.

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.

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

- Bertsimas, Dimitris e Tsitsiklis, John N. (1997). *Introduction to Linear Optimization*. Athena Scientific Belmont, MA.
- 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*. 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.
- Boschetti, Marco A.; Maniezzo, Vittorio; Roffilli, Matteo e Bolufé Röhlér, Antonio. (2009). Matheuristics: Optimization, simulation and control. Blesa, María J.; Blum, Christian; Di Gaspero, Luca; Roli, Andrea; Sampels, Michael e Schaerf, Andrea, editors, *Hybrid Metaheuristics*, p. 171–177, Berlin, Heidelberg. Springer.
- 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.

- Chakraborty, Ripon Kumar; Abbasi, Alireza e Ryan, Michael J. (2020). Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research*, v. 27, n. 1, p. 138–167. doi: 10.1111/itor.12644.
- Changchun, Liu; Xi, Xiang; Canrong, Zhang; Qiang, Wang e Li, Zheng. (2018). A column generation based distributed scheduling algorithm for multi-mode resource constrained project scheduling problem. *Computers & Industrial Engineering*, v. 125, p. 258–278.
- 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.
- 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.
- Demeulemeester, Erik L. (2002). *Project scheduling: a research handbook*, volume 102. Springer.
- 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.
- Fernandes, Gustavo Alves e de Souza, Sérgio Ricardo. (2017). A CLONALG-inspired algorithm with adaptive large neighborhood for the multi-mode resource-constrained project scheduling problem. *Proceedings of the 12th edition of the Metaheuristics International Conference (MIC 2017), Barcelona, July 4-7*, p. 561–570. MIC/MAEB, (2017).

- Fernandes, Gustavo Alves e de Souza, Sérgio Ricardo. (2018). Uma estratégia local branching para o problema de sequenciamento em projetos com restrição em recursos e múltiplos modos de execução. *Proceedings of the Brazilian Symposium on Operations Research (L SBPO), Rio de Janeiro, August 6-9*. SOBRAPO, (2018).
- Fernandes, Gustavo Alves e de Souza, Sérgio Ricardo. (2021). A matheuristic approach to the multi-mode resource constrained project scheduling problem. *Computers & Industrial Engineering*, v. 162, p. 107592.
- Fischetti, Matteo e Lodi, Andrea. (2003). Local branching. *Mathematical programming*, v. 98, n. 1-3, p. 23–47.
- Fréville, Arnaud. (2004). The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research*, v. 155, n. 1, p. 1–21. doi: [https://doi.org/10.1016/S0377-2217\(03\)00274-1](https://doi.org/10.1016/S0377-2217(03)00274-1).
- Geiger, Martin Josef. (2017). A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research*, v. 256, n. 3, p. 729 – 741. doi: <https://doi.org/10.1016/j.ejor.2016.07.024>.
- Gerhards, Patrick; Stuerck, Christian e Fink, Andreas. (2017). An adaptive large neighbourhood search as a matheuristic for the multi-mode resource-constrained project scheduling problem. *European Journal of Industrial Engineering*, v. 11, n. 6, p. 774–791.
- Glover, Fred. (1990). Tabu search-part ii. *ORSA Journal on Computing*, v. 2, n. 1, p. 4–32.
- Glover, Fred e Kochenberger, Gary A. (2003). *Handbook of Metaheuristics*. Springer Science & Business Media.
- 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 Briskorn, Dirk. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, v. 297, n. 1, p. 1–14.
- Hartmann, Sönke e Drexler, 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.

- Hoos, Holger e Stützle, Thomas. (2004). *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. ISBN 1558608729.
- 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 Węglarz, Jan. (2001). Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, v. 102, n. 1-4, p. 137–155.
- Jünger, Michael; Liebling, Thomas M; Naddef, Denis; Nemhauser, George L; Pulleyblank, 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, *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 MRCPS. *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 MRCPS. *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. J., Weglarz, editor, *In: Project scheduling. International Series in Operations Research & Management Science*, volume 14, 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*. Holt, Rinehart and Winston.
- 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.
- Lodi, Andrea. (2009). Mixed integer programming computation. Jünger, Michael; Liebling, Thomas M; Naddef, Denis; Nemhauser, George L; Pulleyblank, William R; Reinelt, Gerhard; Rinaldi, Giovanni e Wolsey, Laurence A, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-art*, Capítulo 16, p. 619–642. Springer Science & Business Media.

- 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.
- López-Ibáñez, Manuel; Dubois-Lacoste, Jérémie; Cáceres, Leslie Pérez; Birattari, Mauro e Stützle, Thomas. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, v. 3, p. 43–58.
- 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.
- 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.
- Machado-Domínguez, Luis F; Paternina-Arboleda, Carlos D; Vélez, Jorge I e Barrios-Sarmiento, Agustin. (2021). A memetic algorithm to address the multi-mode resource-constrained project scheduling problem. *Journal of Scheduling*, v. 24, n. 4, p. 413–429.
- 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; Waligora, Grzegorz e Węglarz, 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.
- Mladenovic, N. e Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097–1100. ISSN 0305-0548. doi: [https://doi.org/10.1016/S0305-0548\(97\)00031-2](https://doi.org/10.1016/S0305-0548(97)00031-2).
- 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. Technical report Tech. Report 3/2011, Department of Management Engineering, Technical University of Denmark. URL

<http://orbit.dtu.dk/ws/files/5704914/rap3saml.2011.pdf>. DTU Management, online: <http://orbit.dtu.dk/ws/files/5704914/rap3saml.2011.pdf>.

Muritiba, Albert Einstein Fernandes; Rodrigues, Carlos Diego e da Costa, Francisco Araujo. (2018). A path-relinking algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research*, v. 92, p. 145–154.

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*. Prentice Hall.

Patterson, J.H.; Slowinski, R.; Talbot, F.B. e Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. Slowinski, R. e Weglarz, J., editors, *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; Walters, 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.

- Raidl, Günther R. e Puchinger, Jakob. (2008). Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. Blum, Christian; Aguilera, Maria José Blesa; Roli, Andrea e Sampels, Michael, editors, *Hybrid Metaheuristics: An Emerging Approach to Optimization*, p. 31–62. Springer, Berlin, Heidelberg. doi: 10.1007/978-3-540-78295-7_2.
- Ramos, Alfredo S; Olivares-Benitez, Elias e Miranda-Gonzalez, Pablo A. (2022). Multi-start iterated local search metaheuristic for the multi-mode resource-constrained project scheduling problem. *Expert Systems*, v. 39, n. 1, p. e12830.
- Santos, Haroldo G.; Soares, Janniele e Tóffolo, Túlio. (2014). Hybrid local search for the multi-mode resource-constrained multi-project scheduling problem. *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling (PATAT'14)*, p. 397–407, (2014).
- Schnell, Alexander e Hartl, Richard F. (2017). On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs. *Operations Research Perspectives*, v. 4, p. 1–11.
- 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 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 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.
- 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 Drexel, 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 Drexel, 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 Drexel, 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.

- 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. (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.
- 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.
- 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. doi: <https://doi.org/10.1016/j.ejor.2010.03.037>.
- 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, C. M. 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.