FORMULAÇÕES E ALGORITMOS SEQUENCIAIS E PARALELOS PARA O PROBLEMA DA ÁRVORE GERADORA DE CUSTO MÍNIMO COM RESTRIÇÃO DE GRAU MÍNIMO

LEONARDO CONEGUNDES MARTINEZ

FORMULAÇÕES E ALGORITMOS SEQUENCIAIS E PARALELOS PARA O PROBLEMA DA ÁRVORE GERADORA DE CUSTO MÍNIMO COM RESTRIÇÃO DE GRAU MÍNIMO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: ALEXANDRE SALLES DA CUNHA

Belo Horizonte

Fevereiro de 2012

Martinez, Leonardo Conegundes
M385f Formulações e algoritmos sequenciais e paralelos para o problema da árvore geradora de custo mínimo com restrição de grau mínimo / Leonardo Conegundes Martinez. — Belo Horizonte, 2012
xxii, 113 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Alexandre Salles da Cunha

1. Otimização combinatória - Teses.
2. Programação (Matemática) - Teses. 3. Programação inteira - Teses. 4. Programação paralela (Computação) - Teses. I. Orientador. II. Título.

CDU 519.6*61 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

O problema da árvore geradora de custo mínimo com restrição de grau mínimo: formulações, algoritmos sequenciais e paralelos

LEONARDO CONEGUNDES MARTINEZ

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ALEXANDRE SALLES DA CUNHA - Orientador
Departamento de Ciência da Computação - UFMG

PROF. GERALDO ROBSON MATEUS
Departamento de Ciência da Computação - UFMG

PROF. MANOEL BEZERRA CAMPÊLO NETO
Departamento de Estatística e Matemática Aplicada - UFC

PROF. MAURÍCIO CARDOSO DE SOUZA
Departamento de Engenharia de Produção - UFMG

Belo Horizonte, 13 de fevereiro de 2012.

Agradecimentos

Aos meus pais e avós, sempre presentes nessa longa caminhada que já dura *exatos* 25 anos! Obrigado por tudo o que fizeram e fazem por mim até hoje. Devo muito dessa conquista a vocês.

Aos meus irmãos e tios(as), muito obrigado pela amizade e apoio contínuo.

À minha amada Débora (uma "exímia cientista da computação!!!"), por todos os momentos que vivemos juntos.

A todo o time da $S^{10}I$ & SI^2, especialmente ao Paulo Gomide, Mateus Lana e Gabriel Lana, que antes de serem sócios, são amigos para a vida inteira!

A todos os que estiveram envolvidos comigo nesses seis anos de Maratona de Programação, em especial aos companheiros de aniMOUSEs e SUDO, Felipe Machado, Itamar Viana e Thiago Goulart. Com vocês resolver problemas e competir sempre foi uma diversão!

A todos os amigos do LaPO, dos velhos e dos novos tempos, em especial ao Fred Quintão, por ter me apontado o "caminho das pedras", e ao Cristiano Arbex e ao Rafael Coelho, que muito me ajudaram a percorrê-lo.

A todos os professores do DCC com quem tive a oportunidade de trocar experiências, discutir ideias e aprender. Em especial ao Prof. Robson, meu primeiro orientador de Iniciação Científica, que contribuiu bastante para a minha formação.

Ao Prof. Alexandre os meus sinceros agradecimentos, por ter sido muito mais que um orientador nos últimos anos. Obrigado por tudo o que fizemos juntos, por sua enorme dedicação, amizade, ensinamentos e apoio ao meu "estilo *multi-thread*".

Resumo

Dados um grafo G não direcionado valorado nas arestas e um inteiro positivo d, o *Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo (PAGMGM)* consiste em encontrar uma árvore geradora de custo mínimo T de G, tal que o grau de cada vértice em T seja igual a 1 ou maior ou igual a d. O PAGMGM foi proposto recentemente e pertence à classe NP-Difícil para $d \geq 3$. Neste trabalho, introduzimos formulações de Programação Inteira e algoritmos de otimização para o problema. Duas formulações baseadas em um número exponencial de restrições de eliminação de subcircuitos, uma direcionada e outra não direcionada, são apresentadas e comparadas sob uma perspectiva teórica e computacional em relação aos seus limites de Programação Linear. Dois métodos baseados na formulação direcionada foram propostos, um algoritmo Branch-and-cut e um método Local Branching. Este último emprega o algoritmo Branch-and-cut como resolvedor interno. Devido ao fato de a formulação direcionada não ser simétrica em relação aos limites de Programação Linear fornecidos, introduzimos uma reformulação compacta e simétrica para o problema, obtida com a aplicação de uma técnica de reformulação por interseção à formulação direcionada. Apesar da reformulação prover limites de Programação Linear muito mais fortes que aqueles fornecidos pelas demais formulações, a avaliação direta desses limites através de resolvedores de Programação Linear é muito cara computacionalmente. Por isso, introduzimos um algoritmo de Relaxação Lagrangeana para aproximá-los. Com o objetivo de acelerar o cálculo dos limites duais Lagrangeanos, implementamos um Método de Subgradiente paralelo. Introduzimos também uma Heurística Lagrangeana baseada no algoritmo Local Branching. Com os métodos propostos, vários novos certificados de otimalidade e melhores limites inferiores e superiores para o PAGMGM são fornecidos.

Palavras-chave: Otimização Combinatória, Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo, Formulações de Programação Inteira, Branch-and-cut, Local Branching, Relaxação Lagrangeana, Programação Paralela.

Abstract

Given an edge weighted undirected graph G and a positive integer d, the *Min-degree Constrained Minimum Spanning Tree Problem (MDMST)* consists of finding a minimum cost spanning tree T of G, such that each vertex is either a leaf or else has a degree at least d in T. The MDMST was recently proposed and was proven to be NP-Hard for $d \geq 3$. In this work, we discuss several formulations and optimization algorithms for the problem. We introduce two Integer Programming formulations based on exponentially many undirected and directed subtour breaking constraints and compare the strength of their Linear Programming bounds, both theoretically and computationally. A Branch-and-cut (BC) algorithm and a Local Branching method that uses BC as its inner solver are proposed. Both methods rely on the directed formulation. Aiming to overcome the fact that the directed formulation is not symmetric with respect to the Linear Programming bounds, we also present a symmetric and compact reformulation, devised with the application of an intersection reformulation technique to the directed model. The reformulation proved to be much stronger than the previous formulations, but evaluating its bounds directly by Linear Programming solvers is very time-consuming. Therefore, we propose a Lagrangean Relaxation algorithm for approximating them. Attempting to speed up the computation of the Lagrangean dual bounds, we implemented a parallel Subgradient Method. We also introduced a Lagrangean Heuristic based on the Local Branching algorithm. With the proposed methods, several new optimality certificates and best lower and upper bounds for MDMST are provided.

Keywords: Combinatorial Optimization, Min-degree Constrained Minimum Spanning Tree Problem, Integer Programming Formulations, Branch-and-cut, Local Branching, Lagrangian Relaxation, Parallel Programming.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

BB	*Branch-and-bound*
BB-MTZ	Algoritmo *Branch-and-bound* para o PAGMGM proposto em [Akgún & Tansel, 2010]
BC	*Branch-and-cut*
CMP	Heurística Construtiva Multipartida Probabilística
DCUT	*Directed Cutset Constraint*
DL	Problema Dual Lagrangeano
DSM	*Deflected Subgradient Method*
ESO	*Enhanced Second Order algorithm*
HL	Heurística Lagrangeana
HVNS	Meta-heurísticas para o PAGMGM propostas em [Martins & Souza, 2009]
LB	*Local Branching*
MIMD	*Multiple-Instruction, Multiple-Data*
MS	Método de Subgradiente
MTZ	Miller-Tucker-Zemlim
PAGMGM	Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo
PAGMG	Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau
PAGM	Problema da Árvore Geradora de Custo Mínimo
PI	Programação Inteira
PL	Programação Linear
RL	Relaxação Lagrangeana
SEC	*Subtour Elimination Constraint*
SIMD	*Single-Instruction, Multiple-Data*
SPMD	*Single Program Multiple Data*
VNS	*Variable Neighborhood Search*

Sumário

Capítulo 1

Introdução

Árvores geradoras são estruturas de fundamental importância em Teoria dos Grafos e no projeto de redes de comunicação cujas entidades (equipamentos, consumidores, fornecedores de serviço, etc.) devem se manter conexas. Assim sendo, um dos problemas mais importantes em Otimização Combinatória consiste em, dado um grafo não direcionado valorado nas arestas, encontrar uma de suas árvores geradoras de custo mínimo. Na literatura, esse problema é usualmente denominado Problema da Árvore Geradora de Custo Mínimo (PAGM).

O PAGM pode ser resolvido eficientemente, por exemplo, através dos algoritmos de Kruskal [1956] e Prim [1957]. Em consonância com esse fato, são conhecidas formulações de Programação Inteira para o problema definidas em termos de Poliedros Inteiros [Edmonds, 1971; Maculan, 1987]. Isso significa dizer que o PAGM também pode ser resolvido por meio de algoritmos baseados em Programação Linear.

Em muitas aplicações, torna-se necessário impor algumas restrições adicionais, para que topologias como árvores geradoras possam ser empregadas no projeto de redes de comunicação. Tais restrições podem, por exemplo, buscar garantir um certo nível de qualidade de serviço de rede, como baixa propensão a falhas. Assim sendo, quando restrições complicantes adicionais são impostas ao PAGM, como limitações no diâmetro da árvore ou no grau de seus vértices, em geral o problema resultante é difícil de ser resolvido. Doze variações do PAGM pertencentes à classe de problemas NP-Difícil foram listadas em [Garey & Johnson, 1979] e vários outros problemas de otimização definidos em árvores com restrições complicantes adicionais foram discutidos em [Deo & Kumar, 1997].

O problema estudado nesta dissertação é conhecido como *Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo* (PAGMGM). Dados um

grafo não direcionado valorado nas arestas e um inteiro positivo d, o objetivo do PAGMGM consiste em encontrar uma árvore geradora de custo mínimo tal que todo vértice na árvore seja uma folha ou tenha grau maior ou igual a d. Dentre os demais problemas definidos em árvores geradoras com restrições complicantes, destaca-se a forte relação do PAGMGM com o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau (PAGMG) [Deo & Hakimi, 1968], para o qual é imposto um limite superior no grau dos vértices de uma árvore geradora viável. Diferentemente do PAGMG, que já foi bastante estudado na literatura [Narula & Ho, 1980; Savelsbergh & Volgenant, 1985; Volgenant, 1989; Caccetta & Hill, 2001; Ribeiro & Souza, 2002; Andrade et al., 2006; Cunha & Lucena, 2007; Souza & Martins, 2008], o PAGMGM foi proposto recentemente por Almeida et al. [2006]. Nesse estudo, o PAGMGM foi demonstrado ser NP-Difícil para $d \geq 4$. Posteriormente, Almeida et al. [2010] provaram que o problema também é NP-Difícil para $d = 3$.

Aplicações do PAGMGM são associadas ao projeto ótimo de redes em que os nós devem estar todos interligados, e recursos, informações ou processamento devem ser centralizados em alguns nós especiais (nós centrais), ao invés de serem dispersos pela rede. Para tanto, cada nó central deve possuir um grau mínimo (representando, por exemplo, o número de clientes que podem ser atendidos ou o número de dispositivos com os quais é possível se comunicar diretamente), de modo a compensar ou tornar úteis os recursos ou facilidades nele instalados. Caso contrário, o nó deve assumir o papel de um vértice folha. Considerando que o custo para conectar qualquer par de nós é não negativo, a solução para problemas nos cenários descritos pode ser representada por uma árvore geradora de custo mínimo, com imposição de um limite inferior para o grau dos vértices centrais.

1.1 Principais Contribuições

As principais contribuições desta dissertação são listadas a seguir na ordem em que aparecem no texto.

- Discussão de três novas formulações de Programação Inteira para o PAGMGM. Duas formulações, uma direcionada e outra não direcionada, são baseadas em um número exponencial de restrições de eliminação de subcircuitos. A terceira formulação é compacta e resulta da aplicação da técnica de Reformulação por Interseção [Gouveia & Telhada, 2008]. Essa reformulação fornece limites de Programação Linear mais fortes que os fornecidos por todas as outras formulações propostas na literatura do PAGMGM.

- Introdução de duas novas classes de desigualdades lineares válidas para o PAGMGM, não redundantes para as formulações conhecidas para o problema.
- Desenvolvimento de algoritmos de otimização sequenciais e paralelos para o PAGMGM, a saber: um algoritmo Branch-and-cut, um método Local Branching que emprega o Branch-and-cut como resolvedor interno e um algoritmo de Relaxação Lagrangeana paralelo. Também são propostas uma heurística construtiva multipartida probabilística, que envolve a resolução de subproblemas de Programação Linear, e uma Heurística Lagrangeana baseada no algoritmo Local Branching.
- Novos certificados de otimalidade e novos melhores limites inferiores e superiores são fornecidos para várias instâncias do PAGMGM.

1.2 Organização da Dissertação

Esta dissertação está estruturada em seis capítulos e dois apêndices. O restante do texto está organizado da seguinte forma:

Capítulo 2. [O Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo] O PAGMGM, suas principais propriedades e aspectos de complexidade computacional são introduzidos. Uma revisão bibliográfica sobre os trabalhos que já abordaram o problema também é apresentada.

Capítulo 3. [Formulações de Programação Inteira para o PAGMGM] Formulações de Programação Inteira para o PAGMGM são introduzidas e comparadas em relação aos seus limites de Programação Linear, sob uma perspectiva teórica e computacional. Algoritmos de planos de corte empregados para avaliar tais limites são apresentados, assim como novas desigualdades lineares válidas para o problema, não redundantes para as formulações da literatura.

Capítulo 4. [Heurística e Algoritmos para a Resolução Exata do PAGMGM] São introduzidos uma heurística construtiva probabilística e dois métodos para a resolução exata do PAGMGM, a saber: um algoritmo Branch-and-cut e um algoritmo Local Branching. Resultados de experimentos computacionais realizados com esses métodos são apresentados e comparados aos resultados obtidos pelos melhores métodos para a resolução do PAGMGM propostos na literatura.

Capítulo 5. [Algoritmos Sequenciais e Paralelos para o PAGMGM Baseados em Relaxação Lagrangeana] Uma Relaxação Lagrangeana baseada em uma reformulação para o PAGMGM é apresentada, juntamente com uma implementação paralela de uma variação do Método de Subgradiente para resolver o Problema Dual Lagrangeano. Uma Heurística Lagrangeana baseada no algoritmo Local Branching é também introduzida. Resultados computacionais obtidos pela Relaxação Lagrangeana são comparados àqueles fornecidos pelos outros métodos avaliados nesta dissertação.

Capítulo 6. [Conclusões e Trabalhos Futuros] Contribuições e limitações desta dissertação são sumarizadas, e finalmente, conclusões e direções para trabalhos futuros são apresentadas.

Apêndice 1. [Matriz de Custo de uma Instância de 10 vértices para o PAGMGM] A matriz de custo de uma instância pequena para o PAGMGM utilizada para a prova de alguns resultados teóricos obtidos nesta dissertação é apresentada.

Apêndice 2. [Compilação dos Melhores Limites Inferiores e Superiores Conhecidos para o PAGMGM] Os melhores resultados conhecidos para todas as instâncias da literatura do PAGMGM são sumarizados.

Capítulo 2

O Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo

Neste capítulo, definimos o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo, e discutimos seus aspectos de complexidade computacional, bem como algumas de suas principais propriedades. Finalizamos o capítulo apresentando uma revisão bibliográfica com os trabalhos que previamente abordaram o PAGMGM, com o intuito de introduzir ao leitor as técnicas já empregadas para tratar o problema.

2.1 O PAGMGM e Suas Propriedades

Seja $G = (V, E)$ um grafo não direcionado, com conjunto de vértices $V = \{1, \ldots, n\}$ e conjunto de arestas E ($m = |E|$). Assuma que custos $\{c_{ij} \in \mathbb{R}_+ : \{i, j\} \in E\}$ são associados às arestas de E. Dado um inteiro positivo $d < n$, o *Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo* consiste em encontrar uma árvore geradora de custo mínimo T de G, tal que o grau de cada vértice em T seja igual a 1 ou maior ou igual a d.

A Figura 2.1 ilustra exemplos de árvores geradoras viáveis para duas instâncias do PAGMGM. A instância cuja solução viável é representada pela árvore da esquerda foi definida em um grafo com $n = 9$ vértices e parâmetro de grau mínimo $d = 3$. Para a instância da direita, temos $n = 10$ e $d = 5$. Em ambas as figuras, os vértices com grau maior ou igual a d (*vértices centrais*) estão destacados em cinza,

enquanto que os vértices com grau igual a 1 (*vértices folha*) são representados na cor branca.

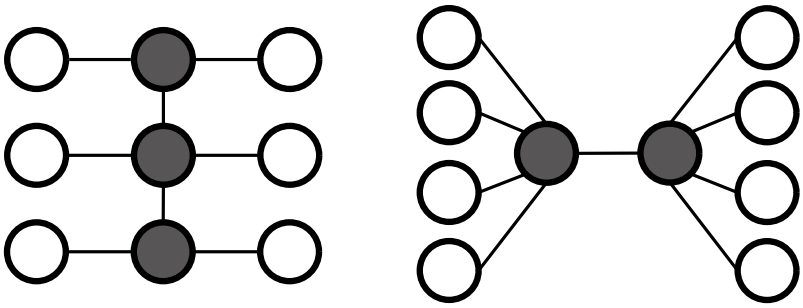

Figura 2.1. Exemplos de árvores geradoras viáveis para o PAGMGM, com parâmetro de grau mínimo $d = 3$ (esquerda) e $d = 5$ (direita). Vértices centrais são destacados na cor cinza.

Almeida et al. [2006] provaram que o PAGMGM é da classe NP-Difícil para $4 \leq d \leq \lfloor n/2 \rfloor$, por meio de uma redução polinomial do Problema do Emparelhamento k-Dimensional [Papadimitriou, 1994] para o PAGMGM. Em um estudo posterior, Almeida et al. [2010] provaram que o problema é também NP-Difícil para o caso em que $d = 3$. Apesar de sua dificuldade no caso geral, existem dois casos particulares do PAGMGM que podem ser resolvidos em tempo polinomial. As Proposições 1 e 2 mostram esses casos.

Proposição 1. *Se $d \leq 2$, então o PAGMGM corresponde ao PAGM e pode ser resolvido em tempo polinomial.*

Prova. Quando $d \leq 2$, o grau de cada vértice em uma árvore viável pode assumir qualquer valor. Nesse caso, o PAGMGM corresponde ao problema clássico da Árvore Geradora Mínima, para o qual são conhecidos diversos algoritmos polinomiais [Kruskal, 1956; Prim, 1957]. □

Proposição 2. *Se $d \geq \lfloor n/2 \rfloor + 1$, então qualquer solução viável para o PAGMGM é uma árvore-estrela e o problema pode ser resolvido em tempo polinomial.*

Prova. Por contradição. Suponha que $d \geq \lfloor n/2 \rfloor + 1$ e que o problema admita uma árvore viável T não estrela. Nesse caso, T deve conter pelo menos dois vértices centrais. Para satisfazer a imposição de grau mínimo, cada um destes vértices deve estar conectado a pelo menos outros $\lfloor n/2 \rfloor$ vértices folha não comuns. Portanto, T deve possuir no mínimo $2 + 2\lfloor n/2 \rfloor > n$ vértices, e não pode ser uma solução viável

para o PAGMGM. Logo, quando $d \geq \lfloor n/2 \rfloor + 1$, todas as soluções viáveis para o PAGMGM são árvores-estrela, e podem ser encontradas por inspeção em $O(n^2)$. □

Outras duas propriedades importantes do PAGMGM foram utilizadas nas formulações e algoritmos propostos nesta dissertação, e são apresentadas nas Proposições 3 e 4. Para tanto, vamos considerar que dados um grafo $G = (V, E)$ e $W \subseteq V$, $E(W) := \{\{i, j\} \in E : i, j \in W\}$. Além disso, considere que $T = (V, E_T)$ denote uma árvore geradora de G que satisfaz as restrições de grau mínimo, $g_T(i)$ denote o grau do vértice $i \in V$ em T, $C \subset V$ denote o conjunto dos vértices centrais de T e $G_C = (C, E(C))$ denote o subgrafo de G induzido por C, ou seja, o subgrafo obtido depois de remover de G todos os vértices de $V \setminus C$ e as arestas incidentes a tais vértices. Analogamente, considere que $T_C = (C, E_T(C))$ denote o subgrafo de T induzido por C. Um resultado trivialmente deduzido em relação à topologia do subgrafo T_C é enunciado pela proposição a seguir.

Proposição 3. *O subgrafo T_C é uma árvore geradora de G_C.*

É importante destacar que o custo da subárvore T_C de uma solução ótima T^* para o PAGMGM não é necessariamente igual ao custo da árvore geradora de custo mínimo de G_C.

Outra propriedade introduzida por Almeida et al. [2006] estabelece um limite superior para o número de vértices centrais de qualquer solução viável para o PAGMGM, definido na proposição a seguir.

Proposição 4. *A cardinalidade do conjunto de vértices centrais C em qualquer solução viável para o PAGMGM é limitada pela expressão $1 \leq |C| \leq \lfloor (n-2)/(d-1) \rfloor$.*

Prova. Pelo conhecido Lema do Aperto de Mãos[1] [West, 2001], temos $\sum_{i \in C} g_T(i) + \sum_{i \in V \setminus C} g_T(i) = 2(n-1)$. Como $g_T(i) = 1$, para todo $i \in V \setminus C$, então $\sum_{i \in C} g_T(i) + n - |C| = 2(n-1)$. Já que todo vértice em C tem grau maior ou igual a d, então $d|C| \leq 2(n-1) - n + |C|$. Reescrevendo, temos $(d-1)|C| \leq n-2$, ou ainda, $|C| \leq (n-2)/(d-1)$. Considerando que $|C|$ é um inteiro e que toda solução viável deve possuir pelo menos um vértice central, podemos estabelecer a expressão

$$1 \leq |C| \leq \left\lfloor \frac{n-2}{d-1} \right\rfloor . \tag{2.1}$$

□

[1] *Handshaking Lemma*

2.2 Revisão Bibliográfica

Embora tenha sido proposto muito recentemente, o PAGMGM já despertou considerável interesse da comunidade acadêmica. O problema foi introduzido por Almeida et al. [2006], em um estudo que apresentou sua classe de complexidade computacional e algumas de suas propriedades. No mesmo trabalho, foram propostas formulações de Programação Inteira (PI), que empregam restrições de balanço de fluxo de uma ou de múltiplas mercadorias. Baseado nas formulações, foi testado um algoritmo Branch-and-bound (BB) para a resolução de instâncias de *benchmark* para o PAGMG [Narula & Ho, 1980; Krishnamoorthy et al., 2001], definidas em grafos com até 50 vértices. Para várias instâncias testadas, o algoritmo BB não foi capaz de fornecer o certificado de otimalidade após 3 horas de execução.

Além da dificuldade teórica associada ao PAGMGM, os resultados computacionais apresentados em [Almeida et al., 2006] indicam que as formulações de fluxo para o PAGMGM fornecem limites de Programação Linear (PL) mais fracos que aqueles dados por formulações equivalentes para o PAGMG. Esse resultado sugere ser mais difícil obter uma boa caracterização da envoltória convexa do PAGMGM que do PAGMG, considerando instâncias de dimensões e classes similares.

Akgún & Tansel [2010] apresentaram uma nova formulação para o PAGMGM baseada nas desigualdades de Miller-Tucker-Zemlin (MTZ) [Miller et al., 1960; Desrochers & Laporte, 1991] para eliminação de subcircuitos. Tais desigualdades foram introduzidas originalmente para o Problema do Caixeiro Viajante [Miller et al., 1960] e garantem, por meio de um número polinomial de restrições, que qualquer subgrafo viável para o problema é conexo e acíclico.

Em geral, formulações que empregam as desigualdades MTZ fornecem limites de PL mais fracos que os fornecidos por formulações de fluxo com múltiplas mercadorias [Langevin et al., 1990; Padberg & Sung, 1991; Desrochers & Laporte, 1991]. No caso particular do PAGMGM, experimentos computacionais relatados em [Akgún & Tansel, 2010] mostraram que, quando ambas as formulações consideram restrições de acoplamento propostas naquele estudo, entre variáveis de seleção de arestas e variáveis de seleção de vértices centrais, a diferença entre os limites de PL fornecidos por essas formulações é reduzida. Os autores também testaram um algoritmo BB utilizando a formulação MTZ que, comparado ao algoritmo BB proposto por Almeida et al. [2006], foi capaz de resolver um conjunto similar de instâncias de teste com menor esforço computacional. Utilizando o mesmo limite de tempo de 3 horas, o algoritmo BB baseado na formulação MTZ resolveu todas as instâncias definidas em grafos com até 30 vértices, mas não conseguiu provar a otimalidade

para algumas instâncias com 50 vértices.

Pela primeira vez na literatura, Martins & Souza [2009] investigaram a aplicação de métodos heurísticos para a resolução do PAGMGM. Os autores apresentaram uma heurística construtiva gulosa, métodos baseados em algoritmos do tipo *second order* [Karnaugh, 1976; Martins, 2007] e heurísticas baseadas em diferentes implementações da meta-heurística *Variable Neighborhood Search* (VNS) [Mladenović & Hansen, 1997].

A heurística construtiva gulosa proposta por Martins & Souza [2009] para o PAGMGM é uma adaptação do algoritmo clássico de Kruskal para o PAGM [Kruskal, 1956]. Tal algoritmo itera sobre o conjunto de arestas do grafo em ordem não decrescente de custos, decidindo a cada iteração se a aresta em avaliação é incluída na solução ótima para o PAGM. A única diferença entre os dois métodos é a forma de avaliação das condições necessárias e suficientes para que uma aresta seja incluída na solução. No algoritmo de Kruskal, uma aresta é selecionada sempre que sua inclusão não forme um ciclo com as arestas já pertencentes à solução. Já na heurística para o PAGMGM, além desta condição, é necessário verificar também se a inclusão da nova aresta não impossibilita a construção de uma árvore geradora em que todos os vértices centrais tenham grau maior ou igual a d. Portanto, a inclusão da aresta se dá sempre que a mesma não forma um ciclo com as arestas já selecionadas e não elimina a possibilidade da heurística encontrar uma árvore geradora viável para o PAGMGM. Caso contrário, a aresta é descartada. Cabe destacar que esta heurística construtiva sempre encontra uma árvore viável para o PAGMGM quando o problema é definido em grafos completos. No entanto, para grafos esparsos, a heurística pode terminar sem encontrar uma árvore viável, ainda que exista alguma.

Os autores testaram também a aplicação direta para o PAGMGM de um algoritmo do tipo *second order* proposto por Karnaugh [1976] e de uma versão aprimorada deste algoritmo (*enhanced second order algorithm, ESO*) proposta por Martins [2007]. O método ESO foi empregado na fase de melhoria de diferentes implementações da meta-heurística VNS. Além destes, foi testado um método que apenas utiliza a heurística construtiva para explorar cada vizinhança na meta-heurística VNS.

Um total de sete heurísticas foram propostas e avaliadas para o PAGMGM em [Martins & Souza, 2009]. Os resultados computacionais mostraram que os métodos VNS randomizados que empregaram o ESO em suas fases de melhoria obtiveram os melhores limites superiores para o problema. Pela primeira vez na literatura, instâncias com mais de 50 vértices foram testadas. Os autores aplicaram suas heurísticas em instâncias com até 500 vértices, estabelecendo parâmetros de comparação para

futuros trabalhos relacionados ao PAGMGM. A julgar pela nossa revisão bibliográfica, esse é o único trabalho publicado até o momento que aborda a aplicação de heurísticas para o problema.

Capítulo 3

Formulações de Programação Inteira para o PAGMGM

Neste capítulo, introduzimos três novas formulações de Programação Inteira para o PAGMGM. As formulações são comparadas em relação aos seus limites de Programação Linear, sob uma perspectiva teórica e computacional. As duas formulações mais fracas, uma não direcionada e outra direcionada, são baseadas em um número exponencial de restrições de eliminação de subcircuitos, enquanto que a mais forte emprega um número polinomial de variáveis e restrições e resulta da aplicação da técnica de Reformulação por Interseção [Gouveia & Telhada, 2008]. Apresentamos algoritmos de planos de corte empregados para avaliar os limites de Programação Linear das formulações propostas, bem como novas desigualdades lineares válidas para o problema, não redundantes para as formulações da literatura.

3.1 Introdução

Em geral, problemas de Programação Inteira podem ser formulados a partir de diferentes escolhas para os conjuntos de variáveis e restrições. No caso de problemas de otimização em grafos, cujas soluções são representadas por árvores geradoras sujeitas a restrições complicantes adicionais, como limitação no diâmetro da árvore ou no grau dos vértices, é usual definir a formulação a partir de dois conjuntos independentes de restrições, aqui chamados de *restrições de topologia de árvore geradora* e *restrições complicantes adicionais*. Esses dois conjuntos de restrições devem garantir, respectivamente, que o subgrafo associado a uma solução viável seja acíclico e conexo e que as restrições complicantes adicionais do problema sejam satisfeitas. Para o caso particular do PAGMGM, o segundo conjunto deve impor as

restrições de grau mínimo dos vértices de uma árvore geradora viável.

Várias abordagens podem ser utilizadas para definir o conjunto de restrições de topologia de árvore geradora [Magnanti & Wolsey, 1995]. Para o PAGMGM, foram apresentadas na literatura formulações baseadas em restrições de balanço de fluxo de uma ou de múltiplas mercadorias [Almeida et al., 2006] e em desigualdades de Miller-Tucker-Zemlin [Akgún & Tansel, 2010]. Uma vantagem das formulações baseadas em fluxos em redes e nas restrições MTZ é o fato de serem compactas, isto é, empregarem um número polinomial de variáveis e restrições. No entanto, o tamanho das formulações com múltiplas mercadorias, que envolvem $O(nm)$ variáveis e restrições, torna-as proibitivas para a aplicação direta em procedimentos de enumeração como o Branch-and-bound, a menos que algum tratamento algorítmico adicional seja empregado, por exemplo, o método de Relaxação Lagrangeana [Fischer, 1981; Guignard, 2003]. Por outro lado, formulações baseadas em fluxos de uma única mercadoria ou nas desigualdades MTZ apresentam em geral limites de Programação Linear fracos [Magnanti & Wolsey, 1995].

Introduzimos neste capítulo três formulações para o PAGMGM, duas baseadas em um número exponencial de restrições de eliminação de subcircuitos e outra baseada na técnica de Reformulação por Interseção. Na apresentação das formulações a seguir, são empregadas as seguintes notações e definições. Dado um grafo não direcionado $G = (V, E)$ e um conjunto não vazio $W \subseteq V$, $\delta(W) := \{\{i, j\} \in E : i \in W, j \notin W\}$ define o conjunto de arestas no corte induzido por W, ou seja, arestas com uma extremidade em W e a outra extremidade em $V \setminus W$, e $E(W) := \{\{i, j\} \in E : i, j \in W\}$ define o conjunto de arestas com ambas as extremidades em W. Da mesma forma, dado um grafo direcionado $D = (V, A)$ com conjunto de arcos A e os conjuntos não vazios $S \subset V$ e $W \subset V$, $(S, W) := \{(i, j) \in A : i \in S, j \in W\}$ define o conjunto de arcos que partem de S para W, $\delta^-(W) := \{(i, j) \in A : i \notin W, j \in W\}$ define o conjunto de arcos que incidem em W a partir de $V \setminus W$, $\delta^+(W) := \{(i, j) \in A : i \in W, j \notin W\}$ define o conjunto de arcos que partem de W para $V \setminus W$ e $A(W) := \{(i, j) \in A : i \in W, j \in W\}$ define o conjunto de arcos com ambas as extremidades em W. Para simplificar, quando $W = \{i\}$, os conjuntos $\delta(\{i\})$, $\delta^-(\{i\})$ e $\delta^+(\{i\})$ são representados respectivamente por $\delta(i)$, $\delta^-(i)$ e $\delta^+(i)$.

Dada uma formulação P para o PAGMGM, assuma que $w(P)$ indique o limite de Programação Linear fornecido por P. Se P é definido em termos de um conjunto estendido de variáveis (x, z), sua projeção no espaço das variáveis z é denotado por $Proj_z(P)$. Considere ainda que $\mathbb{B} := \{0, 1\}$ e que $\mathbb{R}$ denote o conjunto dos números reais. Por fim, para qualquer função real $f : Q \to \mathbb{R}$ definida em um domínio finito

Q e $\overline{Q} \subseteq Q$, defina $f(\overline{Q}) := \sum_{q \in \overline{Q}} f_q$.

3.2 Formulações com um Número Exponencial de Restrições de Eliminação de Subcircuitos

Uma formulação canônica para o PAGMGM pode ser obtida empregando-se variáveis binárias associadas às arestas de E e variáveis binárias para selecionar as folhas de V. Essa formulação emprega as restrições de eliminação de subcircuitos não direcionadas (*Subtour Elimination Constrainsts*, SECs) introduzidas por Dantzig et al. [1954] para o Problema do Caixeiro Viajante [Held & Karp, 1970, 1971]. Mais precisamente, assuma que

$$z_{ij} = \begin{cases} 1, & \text{se a aresta } \{i,j\} \in E \text{ é incluída na árvore,} \\ 0, & \text{caso contrário} \end{cases}$$

e

$$y_i = \begin{cases} 1, & \text{se o vértice } i \in V \text{ é folha na árvore,} \\ 0, & \text{se o vértice } i \in V \text{ é central na árvore.} \end{cases}$$

A *formulação não direcionada* para o PAGMGM é dada por:

$$w = \min \left\{ \sum_{\{i,j\} \in E} c_{ij} z_{ij} : (z,y) \in P_n \cap (\mathbb{B}^m, \mathbb{B}^n) \right\}, \tag{3.1}$$

onde o poliedro $P_n \subset \mathbb{R}^{m+n}$ é definido por:

$$z(E) = n - 1, \tag{3.2}$$

$$z(E(S)) \leq |S| - 1, \ \ \forall S \subset V, \ \ S \neq \emptyset, \tag{3.3}$$

$$z_{ij} \geq 0, \ \ \forall \{i,j\} \in E, \tag{3.4}$$

$$z(\delta(i)) \geq 1 + (d-1)(1-y_i), \ \ \forall i \in V, \tag{3.5}$$

$$z(\delta(i)) \leq 1 + (n-2)(1-y_i), \ \ \forall i \in V, \tag{3.6}$$

$$0 \leq y_i \leq 1, \ \ \forall i \in V. \tag{3.7}$$

As desigualdades SECs (3.3) e as restrições (3.2) e (3.4) formam o conjunto de restrições de topologia de árvore geradora, uma vez que definem a envoltória convexa do Politopo das Árvores Geradoras [Magnanti & Wolsey, 1995]. Já as desigual-

dades (3.5) e (3.6) definem o conjunto de restrições de grau mínimo, e restringem as soluções viáveis do PAGMGM às árvores de G que satisfazem as restrições de grau mínimo dos vértices. Note que quando o vértice i é uma folha ($y_i = 1$), as restrições correspondentes em (3.5) e (3.6) impõem que $z(\delta(i)) = 1$. Quando o vértice i é central ($y_i = 0$), a desigualdade correspondente em (3.5) impõe $z(\delta(i)) \geq d$, enquanto que a restrição em (3.6) para o mesmo vértice i é trivialmente satisfeita.

Outra maneira de formular o PAGMGM consiste em considerar o problema em um grafo direcionado $D = (V, A)$ e modelar o conjunto de soluções viáveis como arborescências em D que satisfazem as restrições de grau mínimo. Para tanto, o grafo direcionado D é obtido através da duplicação de cada aresta $\{i, j\}$ de E em dois arcos (i, j) e (j, i), aos quais são associados custos iguais ao custo da aresta que lhes deu origem. Por uma questão de simplicidade, assuma que c_{ij} também denote o custo do arco (i, j) em $A := \{(i, j) \cup (j, i) : \{i, j\} \in E\}$. Cabe observar que ao tratar o problema a partir de um grafo direcionado, necessitamos escolher um vértice especial $r \in V$, para representar a raiz da arborescência procurada.

Além das variáveis y definidas anteriormente, a formulação direcionada para o PAGMGM pode ser obtida empregando-se variáveis binárias associadas aos arcos de A e um número exponencial de restrições de eliminação de subcircuitos direcionadas (*Directed Cutset Constraints, DCUTs*) [Chopra et al., 1992; Koch & Martin, 1998]. De maneira análoga à formulação anterior, assuma que $y_i = 1$ se o vértice $i \in V$ é folha na *arborescência* ($y_i = 0$ caso contrário) e

$$x_{ij}^r = \begin{cases} 1, & \text{se o arco } (i, j) \in A \text{ é incluído na arborescência,} \\ 0, & \text{caso contrário.} \end{cases}$$

Dada uma raiz r, a *formulação direcionada* para o PAGMGM é definida como:

$$w = \min \left\{ \sum_{(i,j) \in A} c_{ij} x_{ij}^r : (x^r, y) \in P_d^r \cap (\mathbb{B}^{2m}, \mathbb{B}^n) \right\}, \tag{3.8}$$

onde o poliedro $P_d^r \subset \mathbb{R}^{2m+n}$ é definido por:

$$x^r(A) = n - 1, \tag{3.9}$$

$$x^r(\delta^-(i)) = 1, \ \ \forall i \in V \setminus \{r\}, \tag{3.10}$$

$$x^r(\delta^-(W)) \geq 1, \ \ \forall W \subset V \setminus \{r\}, \tag{3.11}$$

$$0 \leq x_{ij}^r \leq 1, \ \ \forall (i, j) \in A, \tag{3.12}$$

$$x^r(\delta^+(r)) \geq 1 + (d-1)(1-y_r), \tag{3.13}$$

$$x^r(\delta^+(r)) \leq 1 + (n-2)(1-y_r), \tag{3.14}$$

$$x^r(\delta^+(i)) \geq (d-1)(1-y_i), \ \ \forall i \in V \setminus \{r\}, \tag{3.15}$$

$$x^r(\delta^+(i)) \leq (n-2)(1-y_i), \ \ \forall i \in V \setminus \{r\}, \tag{3.16}$$

$$0 \leq y_i \leq 1, \ \ \forall i \in V. \tag{3.17}$$

A envoltória convexa do Politopo das Arborescências Geradoras de D, enraizadas em r, é definida pelas DCUTs (3.11) em conjunto com as restrições (3.9) e (3.12) [Magnanti & Wolsey, 1995]. As restrições (3.10), embora naturalmente satisfeitas uma vez que (3.9) e (3.11) são impostas, são mantidas na formulação por razões que ficarão claras na Seção 3.5 e no Capítulo 5, e indicam que deve incidir um arco a todo vértice distinto da raiz.

Para garantir as restrições de grau dos vértices em qualquer arborescência viável, é necessário definir separadamente restrições de grau mínimo para a raiz r (restrições (3.13) e (3.14)) e para os demais vértices (restrições (3.15) e (3.16)), uma vez que a existência de uma raiz em formulações direcionadas implica em importantes diferenças na caracterização dos vértices folhas. Por definição, nenhum arco pode incidir à raiz de uma arborescência orientada para fora deste vértice. Por isso, quando r é folha ($y_r = 1$), as restrições (3.13) e (3.14) garantem que $x^r(\delta^+(r)) = 1$ e, quando r é central ($y_r = 0$), a restrição (3.13) impõe $x^r(\delta^+(r)) \geq d$ enquanto que a desigualdade (3.14) é trivialmente satisfeita. Por outro lado, exatamente um arco incide a todo vértice $i \in V \setminus \{r\}$ em uma arborescência viável. Dessa forma, se i é folha ($y_i = 1$), nenhum arco pode partir de i em uma solução viável, como garantem as restrições (3.15) e (3.16). Se i é central ($y_i = 0$), devem partir de i pelo menos $d-1$ arcos para que as condições de grau mínimo dos vértices sejam atendidas. Este limite inferior é estabelecido pelas restrições (3.15). Já as desigualdades (3.16), estabelecem um limite superior para o número de arcos que partem de $i \in V \setminus \{r\}$ e são trivialmente satisfeitas.

É possível mostrar que a projeção da formulação P_n no espaço das variáveis z é equivalente à formulação correspondente para a versão do problema sem a imposição das restrições de grau mínimo, isto é, o problema da Árvore Geradora de Custo Mínimo. Para tanto, assuma que $P_{arv} \subset \mathbb{R}^m$ denote o Politopo das Árvores Geradoras dado por (3.2)-(3.4).

Proposição 5. $Proj_z(P_n) = P_{arv}$.

Prova. Considere qualquer ponto $(\overline{z}, \overline{y})$ em P_n. Observe que $\overline{z} \in P_{arv}$, uma vez que

P_{arv} é definido a partir das restrições de P_n, e portanto $Proj_z(P_n) \subseteq P_{arv}$.

Por outro lado, dada uma solução $\hat{z} \in P_{arv}$, defina $\hat{\theta}_i = \hat{z}(\delta(i))$, para todo $i \in V$. Por (3.2)-(3.4), temos que $0 \leq \hat{z}_{ij} \leq 1$ e $1 \leq \hat{\theta}_i \leq n-1$, para todo $i \in V$. Assuma que $\hat{y} \in \mathbb{B}^n$ e $\hat{q} \in \mathbb{B}^n$, tal que $\hat{q}_i := 1 - \hat{y}_i$, para todo $i \in V$. As restrições (3.5) e (3.6) podem ser reescritas como $1 + (d-1)\hat{q}_i \leq \hat{\theta}_i \leq 1 + (n-2)\hat{q}_i$, para todo $i \in V$, ou ainda

$$\frac{\hat{\theta}_i - 1}{n-2} \leq \hat{q}_i \leq \frac{\hat{\theta}_i - 1}{d-1}, \ \ \forall i \in V. \tag{3.18}$$

Como $1 \leq \hat{\theta}_i \leq n-1$, para todo $i \in V$, então é possível obter uma solução para as variáveis $\hat{q}_i$ com $0 \leq \hat{q}_i \leq 1$, e consequentemente para $\hat{y}_i$ com $0 \leq \hat{y}_i \leq 1$, para todo $i \in V$, que satisfaça as restrições (3.18), por exemplo, $\hat{q}_i = \frac{\hat{\theta}_i - 1}{n-2}$ para todo $i \in V$. Dessa forma, $(\hat{z}, \hat{y}) \in P_n$, $\hat{z} \in Proj_z(P_n)$ e então $P_{arv} \subseteq Proj_z(P_n)$. Como $Proj_z(P_n) \subseteq P_{arv}$ e $P_{arv} \subseteq Proj_z(P_n)$, segue o resultado. □

Empregando os mesmos argumentos, podemos mostrar que a projeção da formulação P_d^r no espaço das variáveis x^r é equivalente à formulação correspondente para o problema da Arborescência Geradora de Custo Mínimo, que não impõe as restrições de grau mínimo. Para tanto, assuma que $P_{arb} \subset \mathbb{R}^{2m}$ denote o Politopo das Arborescências Geradoras dado por (3.9)-(3.12).

Proposição 6. $Proj_{x^r}(P_d^r) = P_{arb}$.

Prova. A prova é análoga à prova da Proposição 5. Dado um ponto $(\overline{x}^r, \overline{y}) \in P_d^r$, temos que $\overline{x}^r \in P_{arb}$ e portanto $Proj_{x^r}(P_d^r) \subseteq P_{arb}$.

Por outro lado, dada uma solução $\hat{x^r} \in P_{arb}$, defina $\hat{\theta}_i = \hat{x^r}(\delta^+(i))$, para todo $i \in V$. As restrições (3.13)-(3.16) podem ser reescritas como $1 + (d-1)\hat{q}_r \leq \hat{\theta}_r \leq 1 + (n-2)\hat{q}_r$ e $(d-1)\hat{q}_i \leq \hat{\theta}_i \leq (n-2)\hat{q}_i$, para todo $i \in V \setminus \{r\}$, ou ainda

$$\frac{\hat{\theta}_r - 1}{n-2} \leq \hat{q}_r \leq \frac{\hat{\theta}_r - 1}{d-1} \quad \text{e} \quad \frac{\hat{\theta}_i}{n-2} \leq \hat{q}_i \leq \frac{\hat{\theta}_i}{d-1}, \ \ \forall i \in V \setminus \{r\}. \tag{3.19}$$

Por (3.9)-(3.12) temos que $1 \leq \hat{\theta}_r \leq n-1$ e $0 \leq \hat{\theta}_i \leq n-2$, para todo $i \in V \setminus \{r\}$. Então é possível obter uma solução para as variáveis $\hat{q}_i$ com $0 \leq \hat{q}_i \leq 1$, e consequentemente para $\hat{y}_i$ com $0 \leq \hat{y}_i \leq 1$, para todo $i \in V$, que satisfaça as restrições (3.19), por exemplo, $\hat{q}_r = \frac{\hat{\theta}_r - 1}{n-2}$ e $\hat{q}_i = \frac{\hat{\theta}_i}{n-2}$ para todo $i \in V \setminus \{r\}$. Dessa forma, $(\hat{x^r}, \hat{y}) \in P_d^r$, $\hat{x^r} \in Proj_{x^r}(P_d^r)$ e então $P_{arb} \subseteq Proj_{x^r}(P_d^r)$. □

As Proposições 5 e 6 são análogas àquelas apresentadas por Almeida et al. [2006] no contexto das formulações para o PAGMGM baseadas em fluxos em redes

com uma ou múltiplas mercadorias. Como resultado das proposições, temos que os limites de Programação Linear $w(P_n)$ e $w(P_d^r)$ são independentes do parâmetro de grau mínimo d e iguais ao custo da solução ótima do PAGM. Isso implica que o valor das variáveis duais ótimas associadas às restrições (3.5)-(3.6) e (3.13)-(3.16) são sempre iguais a zero nas relaxações lineares de P_n e P_d^r, respectivamente, o que sugere a necessidade de imposição de um conjunto diferente de restrições de grau mínimo ou de outras desigualdades válidas para o problema que fortaleçam tais formulações.

Além disso, é sabido que a formulação P_{arb} para o PAGM é simétrica em relação à escolha da raiz r (veja Magnanti & Wolsey [1995]), isto é, $w(P_{arb})$ independente da raiz r da arborescência. Utilizando esse resultado juntamente com a Proposição 6, podemos concluir que a formulação P_d^r também é simétrica em relação à escolha da raiz, ou seja, $w(P_d^{r_1}) = w(P_d^{r_2})$, para todo $r_1, r_2 \in V$. A partir das observações feitas e das Proposições 5 e 6, trivialmente conclui-se que os limites de Programação Linear de P_d^r e P_n são iguais para todo $r \in V$, como indicado na proposição a seguir.

Proposição 7. $w(P_d^r) = w(P_n)$, *para todo* $r \in V$.

3.2.1 Fortalecendo as Formulações

As formulações P_n e P_d^r podem ser fortalecidas com a adição de desigualdades válidas para o politopo do PAGMGM. Uma maneira de fortalecer as formulações consiste em introduzir desigualdades que impõem condições lógicas associadas à seleção de arestas ou arcos cujas extremidades são vértices folha. Para a formulação não direcionada, tais condições impedem a seleção de uma aresta incidente a dois vértices folha. Analogamente, para a formulação direcionada, tais condições impedem a seleção de um arco que parte de um vértice folha distinto da raiz ou a seleção de um arco que parte da raiz para um vértice folha, no caso em que a raiz também é folha. Mais precisamente, P_n pode ser fortalecida com a introdução das restrições (3.20) e P_d^r com a adição das restrições (3.21) e (3.22):

$$z_{ij} + y_i + y_j \leq 2, \ \ \forall \{i,j\} \in E, \tag{3.20}$$

$$x_{ij}^r + y_i \leq 1, \ \ \forall (i,j) \in A \setminus \delta^+(r), \tag{3.21}$$

$$x_{rj}^r + y_r + y_j \leq 2, \ \ \forall (r,j) \in A. \tag{3.22}$$

Quando as restrições (3.20), (3.21) e (3.22) são consideradas, uma outra de-

sigualdade que pode ser empregada para fortalecer as formulações é baseada na Proposição 4. Ela foi proposta em [Almeida et al., 2006] e estabelece um limite superior para o número de vértices centrais em qualquer solução viável para o problema, como definido a seguir:

$$\sum_{i \in V} (1 - y_i) \leq \left\lfloor \frac{n-2}{d-1} \right\rfloor . \tag{3.23}$$

No restante da dissertação, assuma que P_N denote a interseção do poliedro P_n com as restrições (3.23) e (3.20) e que P_D^r denote a interseção do poliedro P_d^r com as restrições (3.23), (3.21) e (3.22).

Proposição 8. $w(P_D^r) \geq w(P_N)$.

Prova. Seja (x^r, y) pertencente a P_D^r. Defina z tal que $z_{ij} = x_{ij}^r + x_{ji}^r,\ \forall \{i,j\} \in E$. Considere os arcos (i,j) e (j,i) de A tal que $i, j \in V \setminus \{r\}$. Por (3.21), temos $x_{ij}^r + y_i \leq 1$ e $x_{ji}^r + y_j \leq 1$. Somando essas desigualdades temos $x_{ij}^r + x_{ji}^r + y_i + y_j = z_{ij} + y_i + y_j \leq 2$. Logo, como (3.21) e (3.22) implicam em (3.20), mas o contrário não é verdadeiro, $w(P_D^r) \geq w(P_N)$. □

Deve ser destacado que a formulação P_D^r não é simétrica em relação à escolha da raiz, isto é, para duas raízes distintas r_1 e r_2, é possível que $w(P_D^{r_1}) \neq w(P_D^{r_2})$. Por razões similares às apresentadas na discussão da Proposição 7 e na prova da Proposição 8, a formulação de fluxos com múltiplas mercadorias apresentada em Almeida et al. [2006] e a formulação baseada nas desigualdades de Miller-Tucker-Zemlin proposta por Akgún & Tansel [2010], quando fortalecidas por (3.23), (3.21) e (3.22), também não são simétrica em relação à escolha da raiz.

Como é provável que não exista uma política para a escolha da raiz que sempre forneça o limite de Programação Linear mais forte [Almeida et al., 2006; Akgún & Tansel, 2010], aplicamos a técnica de Reformulação por Interseção proposta por Gouveia & Telhada [2008], para definir uma nova reformulação para o PAGMGM com limites simétricos e mais fortes. Essa reformulação é discutida a seguir.

3.3 Reformulação por Interseção

Gouveia & Telhada [2008] propuseram uma técnica de reformulação, denominada *Reformulação por Interseção*, com o objetivo de obter uma formulação de Programação Inteira simétrica em relação à escolha da raiz de uma arborescência viável para o Problema da Árvore de Steiner Multi-Valorado (em tradução livre para *Multi-Weighted Steiner Tree Problem*) [Current et al., 1986]. Essa técnica pode ser empregada

para o PAGMGM, fornecendo uma reformulação simétrica e com limites de Programação Linear mais fortes para o problema.

A principal ideia da técnica de reformulação por interseção, quando aplicada ao PAGMGM, consiste em considerar simultaneamente todas as formulações P_D^r, para todo $r \in V$, de forma a encontrar n arborescências geradoras de D, cada uma enraizada em um vértice diferente de V, tal que, desconsiderando a orientação dos arcos, todas utilizem as mesmas arestas de G. Isso significa dizer que se o arco (i, j) é incluído em uma solução do PAGMGM formulado com $P_D^{r_1}$, um dos arcos (i, j) ou (j, i) deve ser incluído na solução de outra formulação $P_D^{r_2}$, para todo $r_2 \neq r_1$.

Para estabelecer a interseção das n formulações P_D^r, são empregadas as variáveis binárias z_{ij} associadas às arestas de E e apresentadas na descrição da formulação P_n, além de variáveis binárias associadas aos arcos, para cada uma das n arborescências:

$$x_{ij}^r = \begin{cases} 1, & \text{se o arco } (i,j) \in A \text{ é incluído na arborescência enraizada em } r \in V, \\ 0, & \text{caso contrário.} \end{cases}$$

A restrição que estabelece a interseção entre as formulações é definida como:

$$z_{ij} = x_{ij}^r + x_{ji}^r, \ \ \forall r \in V, \ \ \forall \{i, j\} \in E. \tag{3.24}$$

Observe que, devido a (3.24), $z_{ij} = 1$ se o arco $(i, j) \in A$ ou $(j, i) \in A$ é incluído em uma das n arborescências e $z_{ij} = 0$ caso contrário, para toda aresta $\{i, j\} \in E$.

Para uma dada raiz $r \in V$, assuma que o poliedro $P_{Dz}^r \subset \mathbb{R}^{3m+n}$ é dado pela interseção das restrições (3.4), (3.9)-(3.17), (3.23), (3.21), (3.22) e $\{z_{ij} = x_{ij}^r + x_{ji}^r, \ \forall \{i, j\} \in E\}$. Defina $P_I \subset \mathbb{R}^{m(2n+1)+n}$ como $P_I := \bigcap_{r=1}^n P_{Dz}^r$ e considere que $x := (x^1, x^2, \ldots, x^n)$. A reformulação por interseção para o PAGMGM é dada por:

$$w = \min \left\{ \sum_{\{i,j\} \in E} c_{ij} z_{ij} : (z, x, y) \in P_I \cap (\mathbb{B}^m, \mathbb{B}^{2mn}, \mathbb{B}^n) \right\}. \tag{3.25}$$

A proposição a seguir estabelece uma relação entre os limites de PL fornecidos pelas formulações P_I e P_D^r.

Proposição 9. $w(P_I) \geq w(P_D^r), \ \ \forall r \in V.$

Prova. Considere qualquer ponto $(\overline{z}, \overline{x}, \overline{y})$ em P_I. Observe que para qualquer raiz $r \in V$, $(\overline{x}^r, \overline{y}) \in P_D^r$, uma vez que todas as restrições que definem tal politopo são

também impostas na definição de P_I. Devido à (3.24), o custo de $(\overline{z}, \overline{x}, \overline{y})$ pode ser reescrito como $\sum_{\{i,j\}\in E} c_{ij}\overline{z}_{ij} = \sum_{\{i,j\}\in E} c_{ij}(\overline{x}^r_{ij} + \overline{x}^r_{ji}) = \sum_{(i,j)\in A} c_{ij}\overline{x}^r_{ij}$, então P^r_D é uma relaxação de P_I e segue o resultado. □

Cabe destacar que devido às restrições (3.10) e (3.24), todos os conjuntos de DCUTs (3.11) em P_I podem ser substituídos por um único conjunto de SECs (3.3). Analogamente, as restrições de grau mínimo (3.13)-(3.16) e as restrições lógicas (3.22) podem ser respectivamente substituídas por (3.5)-(3.6) e (3.20). Dessa forma, uma representação mais conveniente de P_I pode ser dada pela interseção de (3.2)-(3.7), (3.23), (3.20), (3.24) em conjunto com as restrições (3.9), (3.10), (3.12) e (3.21) impostas para toda raiz $r \in V$. O próximo resultado mostra que uma representação *compacta* para P_I no espaço das variáveis (z, x, y) pode ser obtida.

Proposição 10. *As SECs (3.3) não são necessárias para definir o poliedro P_I, isto é, todas as restrições (3.3) são redundantes para P_I.*

Prova. Por contradição. Assuma que $P_I \neq \emptyset$ e que $(\overline{z}, \overline{x}, \overline{y})$ seja um vetor que satisfaça todas as restrições que definem P_I, exceto alguma restrição (3.3). Se $(\overline{z}, \overline{x}, \overline{y})$ não é viável para P_I, então deve existir um subconjunto de vértices V_1 tal que $\overline{z}(E(V_1)) > |V_1| - 1$. Seja $V_2 = V \setminus V_1$ e escolha, sem perda de generalidade, qualquer vértice $v \in V_1$. Por (3.10), temos que $\overline{x}^v(\delta^-(V_2)) + \overline{x}^v(A(V_2)) = |V_2|$. Então, usando (3.24):

$$\begin{aligned} n-1 = \overline{x}^v(A) &= \overline{x}^v(\delta^-(V_1)) + \overline{x}^v(A(V_1)) + \overline{x}^v(\delta^-(V_2)) + \overline{x}^v(A(V_2)) \\ &= \overline{x}^v(\delta^-(V_1)) + \overline{z}(E(V_1)) + |V_2| \\ &\geq \overline{z}(E(V_1)) + |V_2| \end{aligned}$$

Logo, $\overline{z}(E(V_1)) \leq n-1-|V_2| = |V_1|-1$ e temos uma contradição. Consequentemente, $(\overline{z}, \overline{x}, \overline{y})$ não viola nenhuma restrição de eliminação de subcircuitos não direcionada (3.3), e segue o resultado. □

A Proposição 10 é análoga ao resultado obtido por Gouveia & Telhada [2008] (veja Proposição 3 naquele estudo) para a reformulação por interseção proposta para o Problema da Árvore de Steiner Multi-Valorado. Nesse problema, dado um grafo $G = (V, E)$, o conjunto de vértices V é particionado em dois subconjuntos, os vértices *primários* e *secundários*. A cada aresta $e \in E$, são associados dois enlaces aos quais são atribuídos, respectivamente, custos positivos c^1_e e c^2_e, tal que $c^1_e > c^2_e$, para todo $e \in E$. No máximo um dos dois enlaces associados a cada aresta pode ser selecionado em uma solução viável para o problema. Os enlaces com custos c^1 são chamados de primários enquanto os enlaces com custos c^2 de secundários. O obje-

tivo do problema consiste em encontrar um subgrafo conectado de custo mínimo tal que cada par de vértices primários esteja conectado por um caminho com apenas enlaces primários e os demais pares de vértices também estejam conectados por caminhos envolvendo qualquer tipo de enlaces. Gouveia & Telhada [2008] mostraram que as únicas SECs (3.3) que precisam ser consideradas na formulação são aquelas definidas a partir de subconjuntos que envolvam apenas vértices secundários.

3.4 Novas Desigualdades Lineares Válidas para o PAGMGM

Com o objetivo de fortalecer as formulações apresentadas, introduzimos nessa seção novas desigualdades lineares válidas para o PAGMGM. A primeira desigualdade é definida a partir de três subconjuntos de vértices representados por W, $\overline{W}$ e S, e impõe um limite inferior para o número de arcos que devem partir de S para W, considerando as restrições de grau mínimo do PAGMGM.

Teorema 1. *Dados um grafo direcionado $D = (V, A)$ e uma raiz $r \in V$, assuma que $W \subset V$, $r \in W$, $\overline{W} = V \setminus W$ e $S \subset \overline{W}$, tal que $d|S| \geq |\overline{W}|$. Então:*

$$x^r((S, W)) \geq \left(1 - \sum_{i \in S} y_i\right)(d|S| - |\overline{W}|) - x^r(A(S)) \tag{3.26}$$

é válida para o politopo do PAGMGM.

Prova. A prova é dividida em dois casos.

Caso 1: Se S possui pelo menos um vértice folha, isto é, se $\sum_{i \in S} y_i \geq 1$, a desigualdade (3.26) é trivialmente satisfeita, já que $1 - \sum_{i \in S} y_i \leq 0$, $d|S| - |\overline{W}| \geq 0$ e $x^r_{ij} \geq 0$, para todo $(i, j) \in A$.

Caso 2: Portanto, assumimos que $\sum_{i \in S} y_i = 0$. Nesse caso, a desigualdade (3.26) pode ser reescrita como:

$$x^r((S, W)) \geq d|S| - |\overline{W}| - x^r(A(S)). \tag{3.27}$$

Dessa forma, vamos considerar uma solução viável para o PAGMGM em que S é então um conjunto de vértices centrais. De acordo com (3.10), deve existir exatamente um arco incidente a cada vértice de S. Por isso, para satisfazer as restrições de imposição de grau mínimo (3.15) para os vértices pertencentes a

S, pelo menos $(d-1)|S|$ arcos devem ter origem em S. Destes, no máximo $|\overline{W}| - |S|$ podem ser absorvidos pelos vértices em $\overline{W} \setminus S$ e $x^r(A(S))$ são absorvidos internamente pelos próprios vértices de S. A diferença no número de arcos, $(d-1)|S| - \big(|\overline{W}| - |S|\big) - x^r(A(S)) = d|S| - |\overline{W}| - x^r(A(S))$, deve partir de S para W, e a prova está completa. □

Observação 1. *A desigualdade (3.26) não é redundante para P_D^r e para P_I.*

Prova. Considere a instância para o PAGMGM definida em um grafo completo de 10 vértices, cuja matriz de custos é apresentada no Apêndice A. Considere ainda que $d = 3$ e $r = 7$. Defina os conjuntos $W = \{6, 7, 8, 9, 10\}$, $\overline{W} = \{1, 2, 3, 4, 5\}$ e $S = \{1, 2\}$.

Ilustramos na Figura 3.1 uma solução básica $(\overline{x}^r, \overline{y}) \in P_D^r$ com $\overline{y} = (0{,}5; 0; 0; 0; 1; 1; 1; 1; 1; 1)$. Todo vértice $\{i \in V : y_i = 0\}$ é destacado na figura pela cor cinza escuro. O vértice 1 é indicado pela cor cinza claro (para destacar que $y_1 = 0{,}5$) e os demais vértices são indicados pela cor branca. Omitimos do desenho os arcos $(i, j) \in A$ associados a variáveis $\overline{x}_{ij}^r = 0$, e para os demais, informamos na figura o valor de $\overline{x}_{ij}^r$ ao lado do arco correspondente. Observe que o conjunto S foi destacado e que os vérti

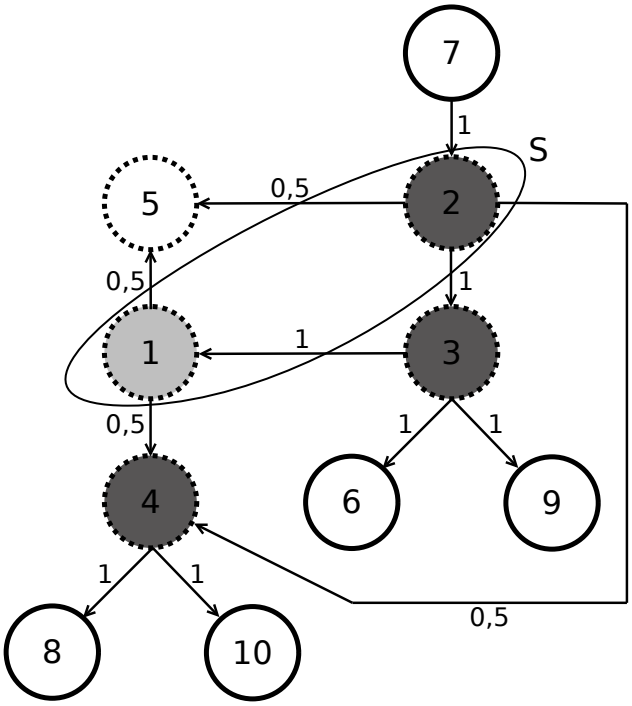

Figura 3.1. Grafo suporte associado a um ponto extremo fracionário de P_D^r que viola a desigualdade (3.26)

Observe que $(\overline{x}^r, \overline{y})$ de fato satisfaz todas as restrições que definem P_D^r, dadas por (3.9)-(3.17), (3.23), (3.21) e (3.22). No entanto, para essa escolha de $\overline{W}$ e S, temos que o lado direito da desigualdade (3.26) é igual a $\left(1 - \sum_{i \in S} y_i\right)\left(d|S| - |\overline{W}|\right) - x^r(A(S)) = 0{,}5 \times (3 \times 2 - 5) - 0 = 0{,}5$, e o lado esquerdo $x^r((S, W)) = 0$, e portanto a desigualdade (3.26) é violada por $(\overline{x}^r, \overline{y})$.

Devido à dificuldade em representar graficamente uma solução para a formulação P_I, sugerimos ao leitor o seguinte procedimento para demonstrar que a desigualdade (3.26) não é redundante para P_I.

Considere a instância cuja matriz de custos está representada no Apêndice A, os parâmetros d e r e os conjuntos W, $\overline{W}$ e S definidos anteriormente. Resolva o seguinte problema de Programação Linear:

$$w = \min \left\{ x^r((S, W)) + \left(\sum_{i \in S} y_i - 1 \right) (d|S| - |\overline{W}|) + x^r(A(S)) : (z, x, y) \in P_I \right\}. \tag{3.28}$$

O custo ótimo de (3.28) é $w = -0{,}5$. Portanto, existe um ponto extremo no poliedro P_I que viola a desigualdade (3.26). □

A segunda desigualdade válida para o PAGMGM introduzida a seguir é baseada na Proposição 3, que mostra que o subgrafo induzido pelos vértices centrais em uma solução viável para o PAGMGM deve ser uma árvore geradora desses vértices.

Teorema 2. *Dados um grafo direcionado $D = (V, A)$ e uma raiz $r \in V$, assuma que $S \subset V$, $S \neq \emptyset$. Então:*

$$x^r(A(S)) \geq \left(\sum_{i \in S} (1 - y_i) + \sum_{i \in V \setminus S} y_i - n + 1 \right) (|S| - 1) \tag{3.29}$$

é válida para o politopo do PAGMGM.

Prova. A prova é dividida em dois casos.

Caso 1: Do total de n vértices do grafo, se existir algum vértice $\{i \in S : y_i = 1\}$ ou $\{i \in V \setminus S : y_i = 0\}$, a desigualdade (3.29) é trivialmente satisfeita, já que $\sum_{i \in S}(1 - y_i) + \sum_{i \in V \setminus S} y_i - n + 1 \leq 0$ e $x_{ij}^r \geq 0$, para todo $(i, j) \in A$.

Caso 2: Portanto, assumimos que $y_i = 0$ para todo $i \in S$ e $y_i = 1$ para todo $i \in V \setminus S$, ou seja, $\sum_{i \in S}(1 - y_i) = |S|$ e $\sum_{i \in V \setminus S} y_i = n - |S|$. Nesse caso,

$\sum_{i \in S}(1 - y_i) + \sum_{i \in V \setminus S} y_i - n + 1 = 1$, e a desigualdade (3.29) pode ser reescrita como:

$$x^r(A(S)) \geq |S| - 1. \tag{3.30}$$

A Proposição 3 mostra que o subgrafo induzido pelos vértices centrais em uma solução viável para o PAGMGM deve induzir uma árvore geradora desses vértices. Como S possui exatamente os vértices centrais, $x^r(A(S)) = |S| - 1$ e a prova está completa.

Observação 2. *A desigualdade (3.29) não é redundante para P_D^r.*

Prova. Considere a instância para o PAGMGM cuja matriz de custos é representada no Apêndice A, $d = 3$, $r = 1$ e uma solução básica $(\overline{x}^r, \overline{y}) \in P_D^r$ com $\overline{y} = (0{,}75; 0; 0; 0; 1; 1; 1; 1; 1; 1)$, obtida em um procedimento análogo ao descrito na Observação 1. Ilustramos a solução $(\overline{x}^r, \overline{y}) \in P_D^r$ na Figura 3.2. Todo vértice $\{i \in V : y_i = 0\}$ é destacado na figura pela cor cinza escuro. O vértice 1 é indicado pela cor cinza claro (para destacar que $y_1 = 0{,}75$) e os demais vértices são indicados pela cor branca. Na solução $(\overline{x}^r, \overline{y})$, $\overline{x}_{ij}^r = 1$ para todo arco $(i, j) \in A$ representado na figura e $\overline{x}_{ij}^r = 0$ para os demais arcos $(i, j) \in A$ omitidos da figura. O valor da variável $\overline{y_i}$ associada ao vértice $i \in V$ é indicado ao lado do vértice na figura. Observe que $(\overline{x}^r, \overline{y})$ de fato satisfaz todas as restrições que definem P_D^r, dadas por (3.9)-(3.17), (3.23), (3.21) e (3.22).

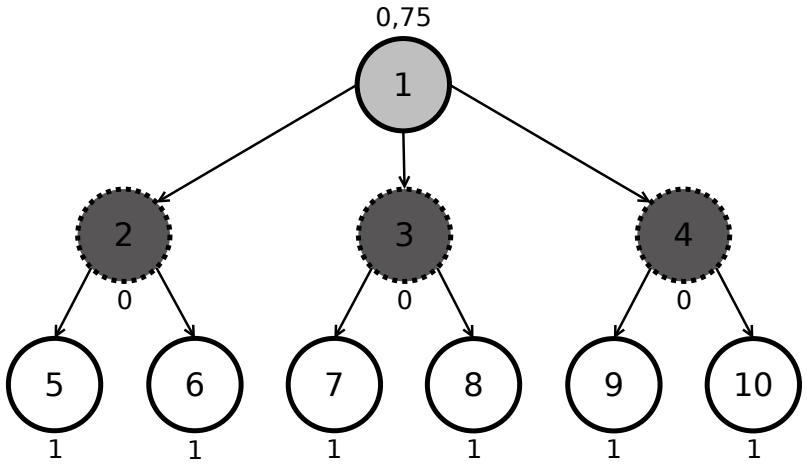

Figura 3.2. Grafo suporte associado a um ponto extremo fracionário de P_D^r que viola a desigualdade (3.29)

Considere o conjunto $S = \{2, 3, 4\}$, cujos vértices foram destacados com contorno tracejado. Para essa escolha de S, temos que o lado direito da desigualdade

(3.29) é igual a $(\sum_{i\in S}(1-y_i)+\sum_{i\in V\setminus S} y_i-n+1)(|S|-1) = (3+6{,}75-10+1)\times(3-1) = 1{,}5$, e o lado esquerdo $x^r(A(S)) = 0$. Portanto, a desigualdade (3.29) é violada por $(\overline{x}^r, \overline{y})$ e a prova está completa. □

Podemos estabelecer uma desigualdade para formulações não direcionadas análoga à desigualdade (3.29), definida no Corolário a seguir.

Corolário 1. *A desigualdade*

$$z(E(S)) \geq \left(\sum_{i\in S}(1-y_i) + \sum_{i\in V\setminus S} y_i - n + 1\right)(|S|-1) \tag{3.31}$$

é válida para o politopo do PAGMGM.

Note que existe um número exponencial de desigualdades (3.26), (3.29) e (3.31). Desse modo, não podemos considerá-las explicitamente nas formulações propostas para o PAGMGM. Dessa forma, uma abordagem a seguir é empregar um procedimento para separá-las e utilizá-lo em um método de planos de corte. No entanto, os procedimentos testados até o momento não foram capazes de encontrar desigualdades válidas violadas em tempos computacionais razoáveis. Por isso, optamos por não considerar essas desigualdades nas implementações avaliadas nesta dissertação. Isso posto, descrevemos a seguir os métodos utilizados para avaliar os limites de PL das cinco formulações de Programação Inteira para o PAGMGM discutidas neste trabalho.

3.5 Avaliação dos Limites de Programação Linear das Formulações

Nesta seção, apresentamos os métodos empregados para avaliar os limites de Programação Linear das formulações P_n, P_N, P_d^r, P_D^r e P_I.

Observe inicialmente que, como resultado das Proposições 5 e 6, os limites de PL das formulações P_n e P_d^r são, respectivamente, iguais aos custos da árvore geradora de custo mínimo de G e da arborescência geradora de custo mínimo de D. Assim, embora P_n e P_d^r sejam definidas a partir de um conjunto com um número exponencial de restrições, não é necessário empregar algoritmos de planos de corte para avaliar $w(P_n)$ e $w(P_d^r)$. Esses limites podem ser avaliados de maneira eficiente, por exemplo, por meio dos algoritmos propostos por Kruskal [1956] e por Edmonds [1967], respectivamente. No entanto, pela Proposição 7, sabemos que

$w(P_d^r) = w(P_n)$, para todo $r \in V$. Portanto, na nossa implementação, utilizamos o algoritmo de Kruskal para avaliar tanto $w(P_n)$ quanto $w(P_d^r)$.

Por outro lado, avaliamos os limites PL das formulações P_N, P_D^r e P_I por meio de métodos de planos de corte, descritos a seguir.

3.5.1 Algoritmos de Planos de Corte

Algoritmos de planos de corte [Dantzig et al., 1954; Kelley, 1960] podem ser empregados para resolver problemas de Programação Linear, em particular aqueles formulados originalmente a partir de um conjunto com um número elevado de restrições. Nesse contexto, a ideia básica desses métodos consiste em resolver uma sequência de problemas relaxados, considerando explicitamente apenas um pequeno subconjunto das desigualdades que definem a formulação do problema. Seja h^* uma solução ótima para o problema relaxado. Se h^* satisfaz todas as desigualdades válidas para a formulação original, então h^* é também uma solução ótima para o problema original. Caso contrário, alguma desigualdade válida para a formulação original e violada por h^* deve ser encontrada e incluída na formulação do problema relaxado, que é então reotimizado na iteração seguinte do algoritmo de planos de corte. Esse processo termina quando nenhuma desigualdade válida para a formulação original é violada por h^*.

Três algoritmos de planos de corte foram implementados e testados neste estudo, cada um com o objetivo de avaliar o limite de Programação Linear de uma das formulações P_N, P_D^r e P_I. As diferenças entre esses métodos são as classes de desigualdades válidas que são separadas e a forma como as separações são realizadas, como veremos a seguir.

3.5.1.1 Avaliação dos Limites de Programação Linear da Formulação Direcionada

Inicialmente, vamos considerar o algoritmo de planos de corte empregado para a avaliação dos limites de PL da formulação P_D^r. O algoritmo começa resolvendo o Problema Linear

$$\min \left\{ \sum_{(i,j)\in A} c_{ij} x_{ij}^r : (x^r, y) \in \overline{P}_D^r \right\}, \tag{3.32}$$

em que o poliedro $\overline{P}_D^r$ é dado por (3.9), (3.10), (3.12)-(3.17), (3.23) e (3.22).

Note que as desigualdades DCUTs (3.11) e as restrições lógicas (3.21) não são inicialmente incluídas em $\overline{P}^r_D$, e portanto, $\overline{P}^r_D$ é uma relaxação de P^r_D. Observe também que, apesar das desigualdades (3.10) serem implicadas por (3.9) e (3.11), optamos por considerá-las explicitamente em $\overline{P}^r_D$, uma vez que as DCUTs foram relaxadas. Dessa forma, o método de planos de corte é capaz de avaliar $w(P^r_D)$ de maneira mais rápida.

Isso posto, para avaliar o limite de PL da formulação P^r_D, são necessários dois procedimentos distintos de separação, um para separar as desigualdades DCUTs (3.11) e outro para separar as desigualdades lógicas (3.21).

A cada iteração do algoritmo de planos de corte implementado, inicialmente são separadas as restrições lógicas (3.21), através de um procedimento simples de inspeção, com complexidade de tempo $O(m)$. Considere que $(\overline{x}^r, \overline{y}) \in \overline{P}^r_D$ é uma solução para (3.32). Para todo arco $(i,j) \in A, i \in V \setminus \{r\}$, se $\overline{x}^r_{ij} + \overline{y}_i > 1$, então a restrição $x^r_{ij} + y_i \leq 1$ é violada pela solução $(\overline{x}^r, \overline{y})$. Em nossa implementação, todas as restrições (3.21) violadas por $(\overline{x}^r, \overline{y})$ são inseridas em $\overline{P}^r_D$.

Apesar de existir um número polinomial de restrições (3.21), experimentos computacionais mostraram que para as instâncias do PAGMGM consideradas neste estudo, separar as restrições dessa classe resulta em um algoritmo de planos de corte com melhores tempos computacionais que o obtido ao introduzi-las inicialmente em $\overline{P}^r_D$.

Após a separação das restrições (3.21), separamos as desigualdades (3.11) da seguinte forma. Seja $(\overline{x}^r, \overline{y}) \in \overline{P}^r_D$ uma solução para (3.32). Se todas as desigualdades (3.11) são satisfeitas por $(\overline{x}^r, \overline{y})$, então $(\overline{x}^r, \overline{y})$ resolve (3.32). Caso contrário, algumas delas são adicionadas em $\overline{P}^r_D$ e (3.32) é reotimizado.

Mais especificamente, para separar as desigualdades (3.11), considere que $\overline{D} = (V, \overline{A})$ é um subgrafo de D definido a partir de $(\overline{x}^r, \overline{y})$, tal que $\overline{A} = \{(i,j) \in A : \overline{x}^r_{ij} > 0\}$. Para decidir se alguma desigualdade (3.11) é violada por $(\overline{x}^r, \overline{y})$, podemos resolver $(n-1)$ problemas de fluxo máximo ou corte mínimo [Ahuja et al., 1993], um para cada par r e $i \in V \setminus \{r\}$ da rede formada por D, com capacidade nos arcos dada por $\{\overline{x}^r_{ij} : (i,j) \in \overline{A}\}$. Assuma que $\delta^-(W)$ denote o corte mínimo que separa $r \in V \setminus W$ e $i \in W$. Se a capacidade desse corte, dada por $\sum_{(p,q)\in\delta^-(W)} \overline{x}^r_{pq}$, for menor que 1, a desigualdade $x^r(\delta^-(W)) \geq 1$ é violada pela solução $(\overline{x}^r, \overline{y})$. Dessa forma, uma ou mais desigualdades (3.11) violadas são inseridas em $\overline{P}^r_d$ e (3.32) é reotimizado na iteração seguinte do algoritmo. A política de seleção das desigualdades violadas que são inseridas em $\overline{P}^r_D$ é discutida mais adiante. A nossa implementação do algoritmo de fluxo máximo tem complexidade de tempo $O(n^3)$. Portanto, o procedimento de separação de DCUTs é realizado em tempo proporcional a $O(n^4)$.

Um importante aspecto na implementação de algoritmos de planos de corte diz respeito à política de escolha dos cortes (desigualdades válidas) violados em cada iteração que serão incluídos no problema relaxado. O procedimento apresentado para a separação das desigualdades (3.11), através da resolução de $n-1$ problemas de fluxo máximo, pode encontrar até $n-1$ cortes violados em cada iteração. Dessa forma, decidir quais desses cortes devem ser incluídos no problema relaxado a ser reotimizado é uma questão importante que pode alterar drasticamente o desempenho do algoritmo de planos de corte. Duas políticas opostas consistem em incluir apenas o corte mais violado ou incluir todos os cortes violados. Decisões intermediárias, que consistem na escolha de um subconjunto próprio dos cortes violados para serem inseridos no problema relaxado, são em geral mais eficientes. Vários autores estudaram essas questões, incluindo Koch & Martin [1998] e Lucena & Resende [2004].

Com o objetivo de manter o problema relaxado com um tamanho razoável e de postergar os efeitos de *tailing off* [Padberg & Rinaldi, 1991], implementamos uma estratégia simples para selecionar as restrições (3.11) que são adicionadas no problema relaxado em cada iteração. Como veremos a seguir, tal estratégia obteve bons resultados na prática. Após a resolução de todos os problemas de fluxo máximo, os cortes violados são normalizados (utilizando a norma Euclidiana) e apenas o corte mais violado, isto é, aquele que tiver a capacidade mínima entre todos os cortes violados (com empates resolvidos arbitrariamente), é adicionado em $\overline{P}_D^r$. Os outros cortes são adicionados apenas quando são suficientemente ortogonais ao corte mais violado. Consideramos dois vetores normalizados suficientemente ortogonais se seu produto interno é menor ou igual a um valor de referência ϵ. Depois de alguns experimentos preliminares com $\epsilon \in \{0{,}01, 0{,}02, \ldots, 0{,}20\}$, escolhemos o valor $\epsilon = 0{,}04$, que resultou em menores tempos computacionais médios para a resolução do nó raiz da árvore de enumeração.

Uma análise experimental da inclusão de cortes ortogonais no algoritmo de planos de corte para a avaliação de $w(P_D^r)$ é apresentada a seguir. As Figuras 3.3, 3.4 e 3.5 representam o crescimento dos limites inferiores de $w(P_D^r)$ durante o algoritmo de planos de corte para três instâncias do PAGMGM, uma para cada classe de instância empregada neste estudo. Em cada gráfico, três curvas são descritas, cada uma associada a uma política de escolha de cortes que são incluídos em $\overline{P_D^r}$: (i) todos os cortes violados, (ii) a estratégia aqui proposta: apenas o corte mais violado e os cortes suficientemente ortogonais a ele, e (iii) apenas o corte mais violado. O eixo vertical das figuras representa a razão entre os limites inferiores do algoritmo de planos de corte em cada instante de tempo e $w(P_D^r)$, e o eixo horizontal representa

o tempo computacional, dado em segundos.

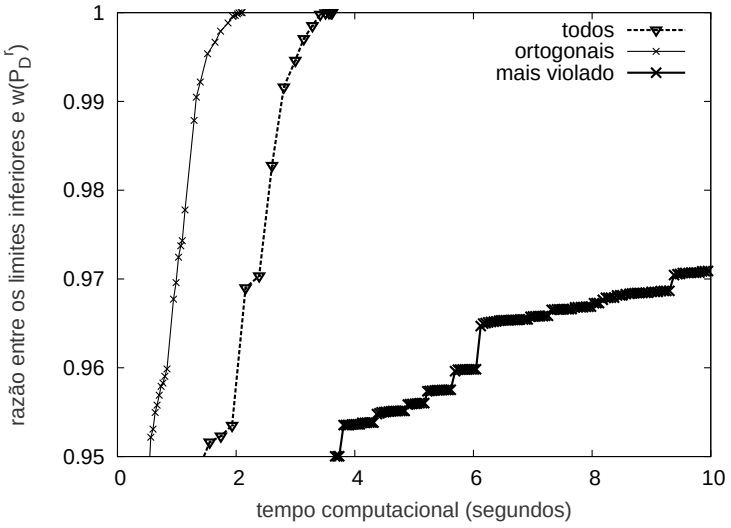

Figura 3.3. Comparação de três políticas de escolha de cortes violados a serem inseridos em $\overline{P_D^r}$. Instância CRD100-3, $d = 5$ (veja Tabela 3.1, id $= 27$).

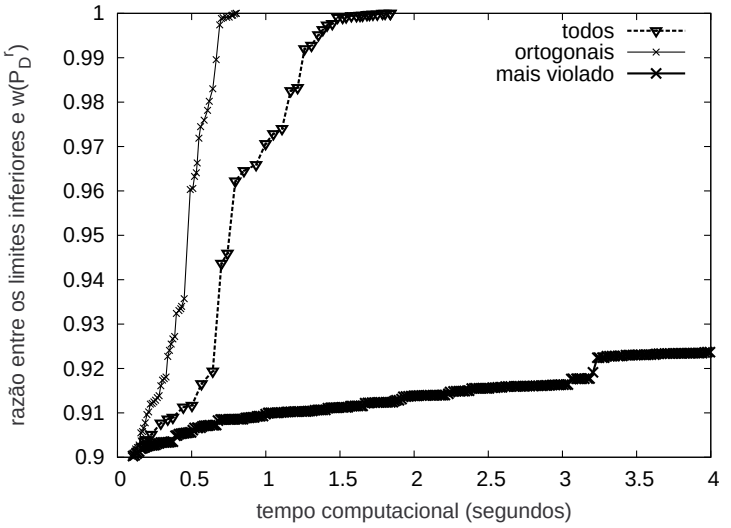

Figura 3.4. Comparação de três políticas de escolha de cortes violados a serem inseridos em $\overline{P_D^r}$. Instância SYM70-3, $d = 3$ (veja Tabela 3.1, id $= 48$).

Os padrões ilustrados nas Figuras 3.3, 3.4 e 3.5 foram observados nas demais instâncias consideradas neste estudo. Note que nos três casos representados pelas figuras, a política de seleção de cortes que adotamos permitiu $w(P_D^r)$ ser avaliado em um menor tempo computacional que aquele necessário pelas duas outras abordagens. Para a instância ALM200-1 (Figura 3.5), por exemplo, o tempo necessário

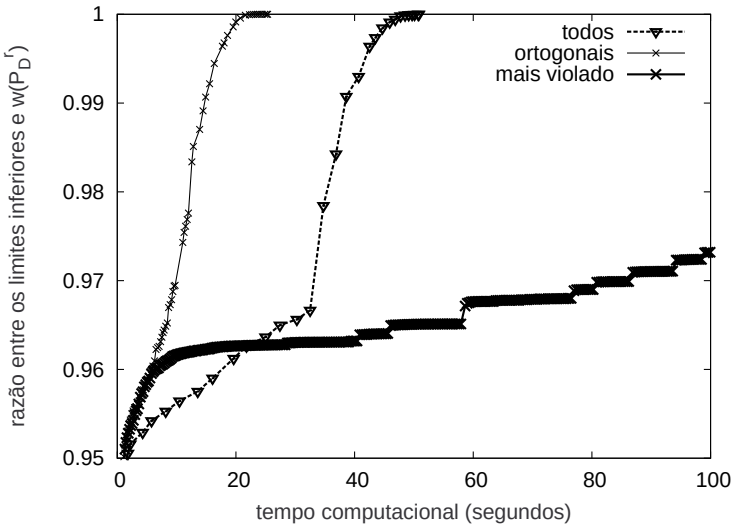

Figura 3.5. Comparação de três políticas de escolha de cortes violados a serem inseridos em $\overline{P_D^r}$. Instância ALM200-1, $d = 5$ (veja Tabela 3.1, id $= 61$).

para avaliar o limite de PL quando a estratégia de cortes ortogonais foi utilizada foi inferior à metade do tempo necessário para avaliar o limite de PL utilizando a estratégia de inclusão de todos os cortes violados.

3.5.1.2 Avaliação dos Limites de Programação Linear da Formulação Não Direcionada

O algoritmo de planos de corte implementado para avaliar $w(P_N)$ funciona de maneira similar ao método apresentado para a avaliação de $w(P_D^r)$. A relaxação de Programação Linear inicial não inclui as SECs (3.3) e as restrições lógicas (3.20). A cada iteração, as restrições (3.20) são separadas por um procedimento simples de inspeção, cuja complexidade de tempo é $O(m)$. As SECs são separadas de maneira exata, empregando o algoritmo proposto por Padberg & Wolsey [1983], que também envolve a resolução de $O(n)$ problemas de fluxo máximo.

Mais precisamente, assuma que $\overline{D} = (\overline{V}, \overline{A})$ denote uma rede capacitada definida a partir de $G = (V, E)$, tal que $\overline{V} = V \cup \{0, n+1\}$ e $\overline{A} = \{(i,j) \cup (j,i) : \{i,j\} \in E\} \cup \{(0,j) : j \in V\} \cup \{(i,n+1) : i \in V\}$. Considere ainda que $(\overline{z}, \overline{y})$ é a solução ótima do problema relaxado, $\overline{a}_i = \overline{z}(\delta(i))$, para todo $i \in V$ e que $\overline{h}_{i,j}$ denote a capacidade do arco $(i,j) \in \overline{A}$, tal que $\overline{h}_{i,j} = \overline{h}_{j,i} = \frac{1}{2}\overline{z}_{ij}$, para toda aresta $\{i,j\} \in E$, $\overline{h}_{0,j} = \max\{\frac{1}{2}\overline{a}_j - 1, 0\}$, para todo arco $(0,j) \in \overline{A}$ e $\overline{h}_{i,n+1} = \max\{1 - \frac{1}{2}\overline{a}_i, 0\}$, para todo arco $(i, n+1) \in \overline{A}$.

Para decidir se alguma SEC (3.3) é violada por $(\overline{z}, \overline{y})$, Padberg & Wolsey [1983]

mostraram que só é necessário resolver $(n-2)$ problemas de fluxo máximo, em redes com capacidades iguais às anteriormente definidas, com a exceção de que no k-ésimo problema, para $k = 1, \ldots, n-2$, deve ser considerada a capacidade $\overline{h}_{0,k} = +\infty$ e, se $k \geq 2$, $\overline{h}_{i,n+1} = +\infty$ para $i = 1, \ldots, k-1$. O k-ésimo problema resulta em um corte da rede capacitada D que separa os vértices 0 e $n+1$, representado por $(S^k \cup \{0\}, \overline{V} \setminus (S^k \cup \{0\}))$. Assim, se $|S^k| - \overline{z}(\delta(S^k)) < 1$, a restrição $z(\delta(S^k)) \leq |S^k| - 1$ é violada pela solução $(\overline{z}, \overline{y})$. Na nossa implementação, o algoritmo de separação das SECs tem complexidade de tempo $O(n^4)$.

O algoritmo de planos de corte implementado para avaliar os limites de PL da formulação P_N também empregou a estratégia de seleção de cortes ortogonais utilizada para avaliar P_D^r.

3.5.1.3 Avaliação dos Limites de Programação Linear da Reformulação por Interseção

Apesar da formulação P_I ser compacta, nossos experimentos computacionais mostraram que o tempo necessário para avaliar $w(P_I)$ é menor quando não incluímos nenhuma restrição lógica (3.21) na relaxação inicial e apenas as adicionamos quando são violadas. Portanto, a avaliação de $w(P_I)$ também foi feita através de um algoritmo de planos de corte, que separa as restrições (3.21) por inspeção, para todo $r \in V$, através de um procedimento de complexidade $O(mn)$.

Uma alternativa para avaliar $w(P_I)$ é o *procedimento iterativo* recentemente proposto por Gouveia & Telhada [2011]. Nesse estudo, os autores apresentaram um método para avaliar os limites inferiores fornecidos por uma formulação (obtida através da aplicação da técnica de reformulação por interseção) para o Problema da Árvore Geradora de Custo Mínimo com Número Mínimo de Folhas. O procedimento iterativo considera inicialmente uma formulação com raiz única e adiciona sequencialmente formulações definidas para outras raízes, impondo a interseção entre todas elas. A ideia do método, quando aplicado para avaliar $w(P_I)$, consiste em obter uma formulação que considera a interseção de um conjunto restrito de formulações P_{Dz}^r, tal que seu limite de PL seja igual ao limite obtido considerando-se desde o início a interseção das formulações para todas as raízes possíveis.

A seguir, descrevemos o algoritmo de planos de corte que emprega o procedimento iterativo proposto em [Gouveia & Telhada, 2011] para avaliar $w(P_I)$. Para tanto, reescrevemos a reformulação por interseção de forma mais conveniente para a apresentação da técnica de Relaxação Lagrangeana no Capítulo 5. Assuma que o poliedro $P_I \subset \mathbb{R}^{m(2n+1)+n}$ seja definido por:

$$\sum_{i \in V}(1 - y_i) \leq \left\lfloor \frac{n-2}{d-1} \right\rfloor, \tag{3.33}$$

$$x^r(A) = n - 1, \ \ \forall r \in V, \tag{3.34}$$

$$x^r(\delta^-(i)) = 1, \ \ \forall r \in V, \ \ i \in V \setminus \{r\}, \tag{3.35}$$

$$(1 - d)y_i - z(\delta(i)) \leq -d, \ \ \forall i \in V, \tag{3.36}$$

$$(n - 2)y_i + z(\delta(i)) \leq n - 1, \ \ \forall i \in V, \tag{3.37}$$

$$z_{ij} - x^r_{ij} - x^r_{ji} = 0, \ \ \forall r \in V, \ \ \{i, j\} \in E, \tag{3.38}$$

$$z_{ij} + y_i + y_j \leq 2, \ \ \forall \{i, j\} \in E, \tag{3.39}$$

$$x^r_{ij} + y_i \leq 1, \ \ \forall r \in V, \ \ (i, j) \in A \setminus \delta^+(r), \tag{3.40}$$

$$z_{ij} \geq 0, \ \ \forall \{i, j\} \in E, \tag{3.41}$$

$$0 \leq y_i \leq 1, \ \ \forall i \in V, \tag{3.42}$$

$$0 \leq x^r_{ij} \leq 1, \ \ \forall r \in V, \ \ (i, j) \in A. \tag{3.43}$$

O procedimento iterativo proposto em [Gouveia & Telhada, 2011] inicia resolvendo o Problema Linear

$$\min \left\{ \sum_{\{i,j\} \in E} c_{ij} z_{ij} : (z, x, y) \in \overline{P}_{It} \right\}, \tag{3.44}$$

em que o poliedro $\overline{P}_{It}$ é dado por (3.33), (3.36), (3.37), (3.41)-(3.43), $\{x^r(\delta^-(i)) = 1, \ \ \forall i \in V \setminus \{r\}\}$ e $\{z_{ij} = x^r_{ij} + x^r_{ji}, \ \ \forall \{i, j\} \in E\}$, com esses dois últimos conjuntos de restrições impostos apenas para um vértice específico $r \in V$. Assim, a formulação relaxada $\overline{P}_{It}$ inclui apenas as restrições de grau-incidente (3.35) e de interseção (3.38) associadas a uma única raiz r, escolhida arbitrariamente (no nosso caso, de acordo com a heurística proposta em [Akgún & Tansel, 2010]). As demais restrições de grau-incidente e de interseção não são incluídas inicialmente em $\overline{P}_{It}$, assim como as restrições lógicas (3.39) e (3.40).

Observe que devido à não inclusão de todas as restrições (3.35) e (3.38) em $\overline{P}_{It}$, a solução ótima de (3.44) pode violar alguma SEC (3.3). No entanto, pela Proposição 10, se uma SEC é violada, o conjunto de vértices que induz essa desigualdade deve envolver apenas os vértices que não foram considerados como raízes na formulação. Então, a cada iteração do algoritmo de planos de corte, o problema (3.44) é resolvido e são determinadas as SECs violadas por sua solução, através do procedimento de

Padberg & Wolsey [1983]. Em seguida, as restrições (3.35) e (3.38) associadas aos vértices pertencentes aos conjuntos que induzem SECs violadas são inseridas em $\overline{P}_{It}$. Por fim, são também incluídas em $\overline{P}_{It}$ todas as restrições lógicas (3.39) e (3.40) violadas pela solução de (3.44).

O limite de PL da formulação P_I é obtido na iteração em que todas as SECs e restrições lógicas são satisfeitas pela solução de (3.44). Neste momento, todas as raízes necessárias para que o limite inferior dado pelo procedimento iterativo seja igual ao fornecido pelo procedimento que considera inicialmente todas as n possíveis raízes terão sido consideradas na formulação. Portanto, através do procedimento iterativo, é possível avaliar $w(P_I)$ sem que todas as n raízes tenham sido consideradas explicitamente.

Implementamos o procedimento iterativo descrito para avaliar $w(P_I)$, e o testamos para as instâncias do PAGMGM utilizadas neste trabalho. Os resultados são apresentados na Seção 3.7.

3.6 Experimentos Computacionais

As duas seções a seguir descrevem o conjunto de instâncias utilizadas para avaliar todos os algoritmos considerados neste trabalho e o ambiente computacional em que os testes foram realizados.

3.6.1 Instâncias de Teste

Utilizamos em nosso trabalho as mesmas classes de instâncias consideradas em [Martins & Souza, 2009], denominadas *CRD*, *SYM* e *ALM*. Todas as instâncias são definidas em grafos completos com custos positivos inteiros associados às arestas. Definimos a seguir essas classes.

CRD: problemas definidos no plano Euclidiano, com coordenadas dos vértices escolhidas de forma aleatória utilizando uma distribuição uniforme em um quadrado. Os custos associados às arestas são dados pela parte inteira da distância Euclidiana entre seus vértices. Foram utilizados problemas definidos em grafos com 30, 50, 70 e 100 vértices.

SYM: problemas similares aos da classe *CRD*, porém definidos em espaços Euclidianos de maiores dimensões. Grafos com 30, 50 e 70 vértices foram testados.

As classes de instâncias *CRD* e *SYM* têm sido utilizadas há vários anos como *benchmark* para o PAGMG [Narula & Ho, 1980; Savelsbergh & Volgenant, 1985; Volgenant, 1989; Krishnamoorthy et al., 2001; Ribeiro & Souza, 2002; Souza & Martins, 2008; Cunha & Lucena, 2007] e foram recentemente empregadas em trabalhos que tratam o PAGMGM [Almeida et al., 2006; Martins & Souza, 2009; Akgún & Tansel, 2010].

No contexto do PAGMG, foram geradas 10 instâncias para cada classe e tamanho de grafo. Seguindo os experimentos computacionais realizados em [Martins & Souza, 2009], utilizamos em nossos testes apenas as três primeiras instâncias de cada classe e tamanho de grafo, totalizando 12 instâncias *CRD* e 9 instâncias *SYM*.

ALM: problemas definidos no plano Euclidiano, com coordenadas dos vértices escolhidas de forma aleatória, utilizando uma distribuição uniforme em um retângulo de dimensões 480 x 640. De forma análoga às instâncias *CRD*, os custos associados às arestas são dados pela parte inteira da distância Euclidiana entre seus vértices. Foram utilizados problemas definidos em grafos com 100, 200, 300, 400 e 500 vértices.

A classe de instâncias *ALM* foi proposta por Andrade et al. [2006] para testar uma heurística Lagrangeana para o PAGMG, tendo se tornado uma importante classe de *benchmark* para o problema [Cunha & Lucena, 2007; Souza & Martins, 2008]. A classe foi também utilizada por Martins & Souza [2009] na avaliação de meta-heurísticas para o PAGMGM. Para cada um dos tamanhos de grafo, três instâncias foram utilizadas em nossos experimentos, totalizando um conjunto de 15 instâncias da classe *ALM*.

Seguindo os experimentos computacionais realizados por Martins & Souza [2009], testamos cada instância com dois ou três valores distintos para o parâmetro de grau mínimo d, com $d \in \{3, 5, 10, 20\}$. Assim, nossos experimentos computacionais utilizaram um conjunto de 90 instâncias, descritas na Tabela 3.1. Cada um dos três blocos de quatro colunas da tabela representa os dados para um conjunto de 30 instâncias. Em cada bloco, as duas primeiras colunas indicam o id e o nome da instância. Os ids são inteiros que escolhemos para referenciar as instâncias no texto, nas tabelas e nos gráficos desta dissertação. Nas duas colunas seguintes, são mostrados o número de vértices da instância e o valor do parâmetro de grau mínimo d utilizado. O primeiro bloco corresponde às instâncias da classe *CRD*. O segundo bloco mostra os dados de 24 instâncias da classe *SYM* (ids 31 a 54) e das 6 menores instâncias da classe *ALM* (ids 55 a 60). Por fim, o terceiro bloco apresenta as infor-

mações das 30 maiores instâncias da classe *ALM*.

Tabela 3.1. Dados das instâncias de teste das classes *CRD*, *SYM* e *ALM* utilizadas nos experimentos computacionais deste trabalho.

Instâncias CRD				Instâncias SYM e ALM				Instâncias ALM			
id	nome	n	d	id	nome	n	d	id	nome	n	d
1	CRD30-1	30	3	31	SYM30-1	30	3	61	ALM200-1	200	5
2	CRD30-2	30	3	32	SYM30-2	30	3	62	ALM200-2	200	5
3	CRD30-3	30	3	33	SYM30-3	30	3	63	ALM200-3	200	5
4	CRD30-1	30	5	34	SYM30-1	30	5	64	ALM200-1	200	10
5	CRD30-2	30	5	35	SYM30-2	30	5	65	ALM200-2	200	10
6	CRD30-3	30	5	36	SYM30-3	30	5	66	ALM200-3	200	10
7	CRD50-1	50	3	37	SYM50-1	50	3	67	ALM300-1	300	5
8	CRD50-2	50	3	38	SYM50-2	50	3	68	ALM300-2	300	5
9	CRD50-3	50	3	39	SYM50-3	50	3	69	ALM300-3	300	5
10	CRD50-1	50	5	40	SYM50-1	50	5	70	ALM300-1	300	10
11	CRD50-2	50	5	41	SYM50-2	50	5	71	ALM300-2	300	10
12	CRD50-3	50	5	42	SYM50-3	50	5	72	ALM300-3	300	10
13	CRD50-1	50	10	43	SYM50-1	50	10	73	ALM400-1	400	5
14	CRD50-2	50	10	44	SYM50-2	50	10	74	ALM400-2	400	5
15	CRD50-3	50	10	45	SYM50-3	50	10	75	ALM400-3	400	5
16	CRD70-1	70	3	46	SYM70-1	70	3	76	ALM400-1	400	10
17	CRD70-2	70	3	47	SYM70-2	70	3	77	ALM400-2	400	10
18	CRD70-3	70	3	48	SYM70-3	70	3	78	ALM400-3	400	10
19	CRD70-1	70	5	49	SYM70-1	70	5	79	ALM400-1	400	20
20	CRD70-2	70	5	50	SYM70-2	70	5	80	ALM400-2	400	20
21	CRD70-3	70	5	51	SYM70-3	70	5	81	ALM400-3	400	20
22	CRD70-1	70	10	52	SYM70-1	70	10	82	ALM500-1	500	5
23	CRD70-2	70	10	53	SYM70-2	70	10	83	ALM500-2	500	5
24	CRD70-3	70	10	54	SYM70-3	70	10	84	ALM500-3	500	5
25	CRD100-1	100	5	55	ALM100-1	100	5	85	ALM500-1	500	10
26	CRD100-2	100	5	56	ALM100-2	100	5	86	ALM500-2	500	10
27	CRD100-3	100	5	57	ALM100-3	100	5	87	ALM500-3	500	10
28	CRD100-1	100	10	58	ALM100-1	100	10	88	ALM500-1	500	20
29	CRD100-2	100	10	59	ALM100-2	100	10	89	ALM500-2	500	20
30	CRD100-3	100	10	60	ALM100-3	100	10	90	ALM500-3	500	20

3.6.2 Ambiente Computacional

Todos os algoritmos propostos nesta dissertação foram implementados em C++ e os experimentos computacionais foram realizados em uma máquina com processador Intel®Core™i7-980 hexa-core, 3,33 GHz, com 16 GB de memória RAM e 12 MB de memória cache L3 compartilhada. O único algoritmo implementado que utiliza os 6 núcleos de processamento em paralelo é a Relaxação Lagrangeana apresentada no Capítulo 5. O compilador g++ versão 4.6.1 foi utilizado, com a *flag* de otimização -O3 ativada.

Alguns algoritmos utilizaram o software CPLEX, versão 10.2 [IBM ILOG CPLEX Optimizer, 2012], através de chamadas à biblioteca Concert Technology, versão 2.4 [ILOG Concert Technology, 2012]. Optamos pelo uso da versão 10.2 do CPLEX em detrimento da versão 12.1 uma vez que a primeira apresentou, na média, menores tempos computacionais para os métodos avaliados neste estudo.

3.7 Resultados Computacionais

Nesta seção, avaliamos computacionalmente os limites de Programação Linear das formulações P_n, P_d^r, P_N, P_D^r e P_I para as instâncias das classes *CRD* e *SYM*, e os comparamos com os limites fornecidos por uma formulação baseada nas desigualdades MTZ introduzida em [Akgún & Tansel, 2010], aqui representados por $w(MTZ)$. Nesse estudo, foram propostas diferentes formulações para o PAGMGM, todas baseadas nas desigualdades MTZ. Consideramos em nossos experimentos computacionais a mais forte dessas formulações.

Em cada instância testada, utilizamos a mesma raiz para definir as formulações P_d^r, P_D^r e MTZ, seguindo a sugestão dada por Akgún & Tansel [2010], que propuseram uma heurística para escolher uma *boa* raiz para formular o PAGMGM em grafos direcionados, descrita a seguir. Seja s_i a soma dos custos dos três arcos mais baratos que partem do vértice i, para todo $i \in V$. A heurística consiste em escolher o vértice i com a menor soma s_i para representar a raiz da arborescência. Empates são resolvidos arbitrariamente. No restante do texto, assumimos que os resultados computacionais associados a algoritmos baseados nas formulações não simétricas para o PAGMGM (P_d^r, P_D^r e MTZ) foram obtidos utilizando essa metodologia para a escolha da raiz.

A avaliação de $w(MTZ)$ foi feita através da resolução da relaxação de Programação Linear da formulação MTZ, utilizando o resolvedor de Programação Linear do CPLEX. Os limites de PL das outras formulações foram avaliados pelos métodos descritos na Seção 3.5. Para cada instância, um limite de tempo de 21600 segundos (6 horas) foi estabelecido para a avaliação dos limites de PL das formulações.

Na Tabela 3.2, apresentamos resultados computacionais detalhados relacionados aos limites de PL fornecidos pelas formulações P_n, P_d^r, MTZ, P_N, P_D^r e P_I para as instâncias das classes *CRD* e *SYM*. Na primeira coluna da tabela é indicado o id de cada instância. A segunda coluna mostra o limite de PL fornecido pelas formulações P_n e P_d^r. Nas colunas seguintes, para cada uma das outras quatro formulações consideradas, é mostrado o seu limite de PL e o tempo necessário t para avaliá-lo, em

segundos. Os tempos gastos para avaliar $w(P_n)$ e $w(P_d^r)$ são desprezíveis, e por isso foram omitidos da tabela. Na última coluna, é indicado o custo da solução ótima de cada instância, representado por w. Para as instâncias em que $w(P_I)$ não pôde ser avaliado no limite de tempo imposto ou o custo da solução ótima não é conhecido, o símbolo "-" é apresentado nas entradas correspondentes da tabela.

Na Tabela 3.3, apresentamos resultados computacionais agregados para grupos formados por instâncias que pertencem à mesma classe e possuem os mesmos valores de n e d. Cada um desses grupos é formado por três instâncias. As primeiras quatro colunas da tabela mostram, respectivamente, a classe, os valores de n e d e os ids das instâncias de cada grupo. Nas colunas seguintes, são indicados, para cada grupo, os *gaps* de dualidade médios em valores percentuais de cada formulação. O *gap* de dualidade entre o limite de PL $w(P)$ de uma formulação P e o custo da solução ótima w é dado por $100(w - w(P))/w$. De maneira análoga à convenção adotada na apresentação da Tabela 3.2, quando $w(P_I)$ não pôde ser avaliado no limite de tempo imposto ou quando o custo da solução ótima não é conhecido para as instâncias de um grupo, o símbolo "-" é mostrado nas entradas correspondentes da tabela.

Os resultados computacionais mostrados na Tabela 3.2 comprovam, como esperado, o resultado teórico resultante das Proposições 5 e 6. Observe que $w(P_n)$ e $w(P_d^r)$ são invariantes em relação ao parâmetro de grau mínimo d e são sempre inteiros, iguais ao custo da solução ótima do PAGM. Isso justifica o fato dessas formulações fornecerem os piores limites de PL apresentados neste estudo. No entanto, note que a adição das restrições (3.23) e (3.20) na formulação P_n e das restrições (3.23), (3.21) e (3.22) em P_d^r resulta em formulações com limites de PL muito mais fortes e sensíveis ao parâmetro d. Em média, $w(P_N)$ e $w(P_D^r)$ foram $56{,}51\%$ e $62{,}06\%$ superiores a $w(P_n)$ e $w(P_d^r)$, respectivamente.

Como pode ser observado pelos resultados, não existe dominância entre $w(MTZ)$ e $w(P_N)$ para as instâncias da classe *CRD*. No entanto, considerando apenas as instâncias da classe *SYM*, $w(MTZ)$ foi mais forte que $w(P_N)$ em todos os casos. Em relação à formulação P_D^r, destacamos que $w(P_D^r)$ foi em média $2{,}3\%$ mais forte que $w(MTZ)$ e $3{,}2\%$ mais forte que $w(P_N)$. Observe que de fato a formulação P_I é muito mais forte que as outras. Na média, $w(P_I)$ foi $6{,}14\%$, $6{,}89\%$ e $3{,}57\%$ superior a $w(MTZ)$, $w(P_N)$ e $w(P_D^r)$, respectivamente. Além disso, em cinco casos $w(P_I)$ foi igual ao custo da solução ótima, e em todos eles as soluções de Programação Linear foram inteiras.

Para os tamanhos de instâncias reportadas na Tabela 3.2, os tempos computacionais necessários para avaliar os limites de PL de todas as formulações conside-

radas, com exceção da P_I, foram baixos e apresentaram a mesma ordem de magnitude. Já os limites de PL fornecidos por P_I, $w(P_I)$, se mostraram muito mais difíceis de serem determinados. Para 12 das 54 instâncias, $w(P_I)$ não pôde ser calculado no limite de tempo imposto. Além disso, para várias instâncias, os tempos computacionais necessários para avaliar $w(P_I)$ foram até quatro ordens de magnitude maiores que aqueles necessários para a avaliação dos limites das outras formulações.

Considerando apenas as instâncias para as quais foi possível avaliar $w(P_I)$, os *gaps* de dualidade médios das formulações MTZ, P_N, P_D^r e P_I foram iguais a $8{,}12\%$, $8{,}82\%$, $5{,}92\%$ e $2{,}81\%$, respectivamente. Se considerarmos as 51 instâncias para as quais é conhecido o custo da solução ótima, os *gaps* de dualidade médios das formulações MTZ, P_N e P_D^r foram iguais a $8{,}13\%$, $9{,}19\%$ e $6{,}24\%$.

Os resultados da Tabela 3.3 mostram *gaps* de dualidade bastante elevados para as formulações P_n e P_d^r, principalmente para valores maiores de d. Observe que para as duas classes de instâncias e para todos os tamanhos de grafos considerados, à medida que o valor de d aumenta, os limites dados pelas formulações P_n e P_d^r se tornam mais fracos. Esse comportamento não é observado para as outras formulações considerando a classe de instâncias *CRD*. No entanto, ao analisarmos apenas as instâncias da classe *SYM* consideradas, percebemos que para todas as formulações e tamanhos de grafos, quanto maior o valor do parâmetro de grau mínimo d, maiores também são os *gaps* de dualidade fornecidos pelas formulações.

A seguir, avaliamos computacionalmente a nossa implementação do procedimento iterativo proposto em [Gouveia & Telhada, 2011] para avaliar o limite de PL da formulação P_I. A Tabela 3.4 compara os desempenhos do procedimento iterativo e do procedimento que considera simultaneamente todas as n raízes desde o início, para as instâncias definidas em grafos com 30 e 50 vértices. O bloco à esquerda da tabela apresenta os resultados associados às instâncias da classe *CRD*, enquanto o bloco à direita corresponde às instâncias da classe *SYM*. Em cada bloco, a primeira coluna mostra os ids das instâncias consideradas. Nas duas colunas seguintes são apresentados os tempos computacionais (em segundos) gastos para a avaliação de $w(P_I)$, empregando o procedimento que considera as n raízes simultaneamente desde o início (coluna t_p), e o procedimento iterativo (coluna t_i). A quarta coluna indica o número total de raízes consideradas pelo procedimento iterativo, ou seja, o número de raízes distintas associadas às restrições (5.3) e (5.6) introduzidas na formulação. Por fim, a última coluna mostra a razão entre os tempos computacionais dos dois métodos.

A Tabela 3.4 mostra que para as instâncias de 30 vértices da classe *CRD*, os

tempos computacionais dos dois métodos foram bastante próximos. No entanto, para todas as outras instâncias, os tempos computacionais gastos pelo procedimento iterativo foram maiores que os tempos gastos pelo procedimento que considera as n raízes simultaneamente, principalmente quando avaliamos as instâncias da classe *SYM*. Por exemplo, para as instâncias com 50 vértices dessa classe, o tempo gasto pelo procedimento iterativo para avaliar $w(P_I)$ foi em média $4{,}88$ vezes o tempo gasto pelo procedimento que considera as n raízes desde o princípio.

Para as instâncias com mais de 50 vértices consideradas neste estudo, o desempenho do procedimento iterativo foi ainda pior. Possíveis explicações para isso são a necessidade extra de resolver $O(n)$ algoritmos de fluxo máximo a cada iteração do algoritmo de planos de corte e a inclusão das restrições (5.3) e (5.6) associadas a todas as n raízes na formulação relaxada para grande parte das instâncias testadas (veja as colunas #raízes na Tabela 3.4). Uma outra possível explicação para o fraco desempenho do procedimento iterativo para avaliar $w(P_I)$ é que restrições de igualdade são incluídas na formulação a cada iteração, o que pode levar à inexistência de uma solução dual viável básica prontamente disponível para reotimização pelo método dual simplex.

Tabela 3.2. Comparação dos limites de Programação Linear dados pelas formulações P_n, P_d^r, MTZ, P_N, P_D^r e P_I, para as instâncias das classes *CRD* e *SYM*.

	P_n, P_d^r	MTZ		P_N		P_D^r		P_I		
id	$w(P_n), w(P_d^r)$	$w(MTZ)$	t	$w(P_N)$	t	$w(P_D^r)$	t	$w(P_I)$	t	w
1	3634	3684,0	0,0	3651,0	0,0	3819,9	0,0	3964,9	23,6	4026
2	3277	3085,5	0,0	3452,0	0,0	3605,4	0,0	3632,5	6,4	3793
3	4001	3889,0	0,0	4014,0	0,0	4076,0	0,0	4166,5	3,7	4293
4	3634	4591,6	0,1	4588,8	0,0	4724,1	0,0	4885,8	47,7	5026
5	3277	4101,4	0,1	4313,4	0,1	4428,5	0,0	4549,0	34,2	4648
6	4001	4914,2	0,0	4940,9	0,0	5012,8	0,0	5209,1	56,5	5425
7	4931	4852,2	0,1	5073,5	0,1	5251,2	0,2	5435,7	243,9	5512
8	5126	5276,0	0,2	5362,0	0,1	5466,9	0,2	5696,1	390,8	5813
9	4898	5159,0	0,1	5289,7	0,1	5394,5	0,2	5540,7	436,8	5590
10	4931	6048,5	0,2	6205,7	0,3	6336,7	0,2	6571,7	1349,1	6908
11	5126	6648,8	0,2	6655,3	0,3	6789,9	0,1	7027,2	2054,6	7204
12	4898	6567,4	0,2	6604,0	0,4	6725,9	0,1	6996,0	1893,1	7277
13	4931	8986,2	0,4	8871,5	0,4	9118,5	0,4	9326,1	6298,6	9633
14	5126	9368,2	0,4	9240,4	0,2	9404,2	0,2	9557,8	4979,4	9743
15	4898	9364,8	0,4	9306,7	0,2	9435,1	0,2	9643,4	4932,4	9855
16	5789	5773,0	0,3	6023,0	0,9	6134,5	0,7	6377,2	2869,7	6516
17	5848	5645,2	0,3	6019,6	0,5	6111,8	0,8	6420,5	2836,2	6586
18	6167	6099,5	0,4	6314,5	0,4	6459,5	1,1	6807,3	2985,8	7053
19	5789	7162,6	0,5	7313,7	1,1	7442,1	0,6	7753,3	16114,6	8134
20	5848	7070,5	0,6	7263,0	1,1	7412,8	0,7	7708,9	15341,5	7943
21	6167	7539,5	0,6	7633,6	1,4	7779,7	0,4	8153,2	8898,2	8419
22	5789	10670,8	0,5	10574,6	1,5	10719,2	0,8	-	-	11235
23	5848	10718,3	0,5	10603,2	0,8	10800,5	0,8	-	-	11373
24	6167	11631,4	0,6	11274,0	3,4	11652,6	0,6	-	-	11979
25	6194	8108,9	0,6	8275,3	5,5	8422,2	1,8	-	-	-
26	7004	8504,9	1,0	8636,9	7,5	8817,5	1,5	-	-	-
27	6831	8173,5	0,7	8555,7	4,9	8697,2	2,1	-	-	-
28	6194	12305,4	1,8	12107,9	8,2	12389,0	2,8	-	-	12916
29	7004	12490,6	1,8	12194,1	10,2	12606,4	2,7	-	-	13026
30	6831	12729,1	1,5	12489,3	4,9	12823,2	3,1	-	-	13365
31	958	1133,2	0,0	1072,3	0,0	1135,4	0,1	1194,0	6,0	1197
32	1219	1395,0	0,0	1373,0	0,0	1395,3	0,0	1435,0	2,6	1435
33	1252	1406,0	0,0	1340,8	0,0	1408,0	0,0	1408,0	4,1	1408
34	958	1653,0	0,0	1594,3	0,0	1658,4	0,0	1765,0	35,8	1765
35	1219	1944,1	0,0	1889,6	0,0	1966,7	0,0	2013,5	31,4	2090
36	1252	1950,5	0,0	1835,7	0,0	1950,5	0,0	2008,0	22,3	2008
37	1098	1233,0	0,1	1190,3	0,0	1233,5	0,1	1274,7	182,3	1278
38	1045	1139,0	0,1	1097,3	0,1	1142,8	0,2	1176,5	275,4	1178
39	1416	1581,8	0,1	1581,5	0,0	1589,0	0,1	1597,5	60,2	1615
40	1098	1831,6	0,1	1774,4	0,0	1831,6	0,1	1954,3	1106,0	2054
41	1045	1669,3	0,2	1580,8	0,1	1675,5	0,1	1741,8	788,8	1760
42	1416	2306,7	0,2	2258,1	0,1	2324,2	0,1	2442,0	964,9	2525
43	1098	3734,8	0,2	3534,0	0,1	3734,8	0,2	3812,8	2605,3	4121
44	1045	3676,5	0,2	3526,2	0,2	3684,1	0,2	3720,9	1672,5	4166
45	1416	4377,6	0,3	4145,5	0,1	4377,6	0,2	4521,3	3918,0	4979
46	1177	1302,0	0,3	1272,0	0,2	1306,2	0,6	1359,0	981,8	1360
47	1185	1339,5	0,2	1301,5	0,0	1351,6	0,6	1433,3	1665,0	1448
48	1232	1446,6	0,2	1418,5	0,2	1450,8	0,8	1521,0	865,7	1521
49	1177	1854,8	0,4	1747,1	0,3	1857,7	0,3	1974,7	13062,2	2028
50	1185	2060,9	0,5	1965,5	0,4	2065,4	0,2	2126,4	7031,0	2165
51	1232	2068,4	0,4	2005,7	0,2	2068,4	0,3	2170,6	8820,3	2210
52	1177	4231,7	0,9	3951,6	0,8	4231,7	0,5	-	-	4979
53	1185	4044,0	0,7	3826,2	0,5	4057,4	0,5	-	-	4787
54	1232	4208,8	0,8	4003,2	0,6	4208,8	0,5	-	-	4997

Tabela 3.3. Comparação dos *gaps* médios entre os limites de Programação Linear dados pelas formulações P_n, P_d^r, MTZ, P_N, P_D^r e P_I e os custos das soluções ótimas das instâncias das classes *CRD* e *SYM*. Resultados calculados considerando a média por classe de instância, n e d.

Instâncias				$w(P_n), w(P_d^r)$	$w(MTZ)$	$w(P_N)$	$w(P_D^r)$	$w(P_I)$
classe	n	d	ids	gap (%)	gap (%)	gap (%)	gap (%)	gap (%)
CRD	30	3	1 – 3	10.05	12.19	8.27	5.04	2.90
		5	4 – 6	27.81	9.94	8.27	6.11	2.97
	50	3	7 – 9	11.58	9.64	7.03	4.73	1.43
		5	10 – 12	30.05	9.97	9.01	7.20	3.73
		10	13 – 15	48.83	5.18	6.21	4.36	2.41
	70	3	16 – 18	11.64	13.07	8.88	7.16	2.71
		5	19 – 21	27.32	11.12	9.33	7.59	3.60
		10	22 – 24	48.52	4.56	6.18	4.12	-
	100	5	25 – 27	-	-	-	-	-
		10	28 – 30	49.05	4.53	6.40	3.79	-
SYM	30	3	31 – 33	15.37	2.75	6.51	2.64	0.08
		5	34 – 36	41.68	5.40	9.28	4.94	1.22
	50	3	37 – 39	12.57	2.96	5.26	2.69	0.49
		5	40 – 42	43.70	8.21	11.46	7.86	3.06
		10	43 – 45	73.28	11.07	15.45	11.01	9.12
	70	3	46 – 48	16.87	5.55	7.78	5.07	0.36
		5	49 – 51	43.83	6.59	10.77	6.47	2.06
		10	52 – 54	75.65	15.43	20.20	15.34	-

Tabela 3.4. Comparação do tempo computacional gasto para avaliar $w(P_I)$ através do procedimento iterativo proposto em [Gouveia & Telhada, 2011] e do procedimento padrão, que emprega todas as n raízes possíveis desde o início.

Instâncias CRD					Instâncias SYM				
id	t_p	t_i	#raízes	$\frac{t_i}{t_p}$	id	t_p	t_i	#raízes	$\frac{t_i}{t_p}$
1	23,6	10,4	30	0,44	31	6,0	10,7	30	1,78
2	6,4	7,2	30	1,12	32	2,6	5,8	22	2,26
3	3,7	4,2	24	1,12	33	4,1	4,3	26	1,05
4	47,7	43,0	30	0,90	34	35,8	107,0	30	2,99
5	34,2	30,5	30	0,89	35	31,4	68,2	30	2,17
6	56,5	52,6	30	0,93	36	22,3	56,6	30	2,54
7	243,9	358,5	50	1,47	37	182,3	681,9	50	3,74
8	390,8	650,6	50	1,66	38	275,4	742,8	50	2,70
9	436,8	535,0	50	1,22	39	60,2	180,8	49	3,01
10	1349,1	2080,1	50	1,54	40	1106,0	5416,5	50	4,90
11	2054,6	3035,9	50	1,48	41	788,8	4486,8	50	5,69
12	1893,1	2506,6	50	1,32	42	964,9	3860,8	50	4,00
13	6298,6	10262,7	50	1,63	43	2605,3	15908,2	50	6,11
14	4979,4	10688,5	50	2,15	44	1672,5	16996,5	50	10,16
15	4932,4	9467,5	50	1,92	45	3918,0	14033,0	50	3,58

3.8 Comentários

Neste capítulo, introduzimos novas formulações de Programação Inteira para o PAGMGM, e as comparamos em relação aos seus limites de relaxação de Programação Linear sob uma perspectiva teórica e computacional. Apesar da formulação P_I fornecer melhores limites inferiores que a formulação P_D^r, na prática, os tempos computacionais gastos para avaliar esses limites, por meio do algoritmo de planos de corte que propusemos ou pelo procedimento iterativo introduzido em [Gouveia & Telhada, 2011], tornam proibitivo o uso da reformulação por interseção em um procedimento enumerativo inteligente, do tipo Branch-and-bound, se nenhum tratamento adicional for feito.

Em uma primeira tentativa de resolver o PAGMGM, introduzimos no próximo capítulo dois algoritmos de resolução exata baseados na formulação P_D^r, a saber: um algoritmo Branch-and-cut e um algoritmo Local Branching. Esses métodos foram capazes de fornecer novos certificados de otimalidade e obter melhores limites inferiores e superiores para várias instâncias do PAGMGM.

Capítulo 4

Heurística e Algoritmos para a Resolução Exata do PAGMGM

Neste capítulo, introduzimos uma heurística e dois métodos para a resolução exata do PAGMGM, a saber: um algoritmo Branch-and-cut e um algoritmo Local Branching. Os métodos exatos são baseados na formulação P_D^r, e utilizam a heurística probabilística para gerar uma solução inicial viável para o problema. Com esses algoritmos, pretendemos resolver o PAGMGM com um esforço computacional inferior ao requerido pelos demais métodos propostos para o problema, assim como fornecer melhores limites inferiores e superiores para as instâncias do PAGMGM cujo certificado de otimalidade não é conhecido. Ao final do capítulo, comparamos os resultados obtidos pelos dois algoritmos com os fornecidos pelos melhores métodos para a resolução do PAGMGM da literatura, propostos por Martins & Souza [2009] e Akgún & Tansel [2010].

4.1 Uma Heurística Construtiva Multipartida Probabilística

Introduzimos nesta seção uma heurística construtiva multipartida probabilística para o PAGMGM, denominada CMP, cujo objetivo consiste em encontrar de maneira rápida uma solução de boa qualidade para o problema. A heurística envolve duas fases: (i) escolha de um conjunto de vértices centrais e (ii) escolha das arestas da árvore geradora baseada na escolha particular dos vértices centrais. A fase (i) é resolvida através de um procedimento probabilístico. Na fase (ii), as arestas cujas duas extremidades são vértices centrais são selecionadas por uma vari-

ação do algoritmo de Kruskal para o PAGM [Kruskal, 1956], enquanto a seleção das arestas que conectam as folhas aos vértices centrais é feita através da resolução de um problema de Programação Linear convenientemente formulado.

A heurística CMP funciona da seguinte forma. Com o objetivo de encontrar uma solução viável $(\hat{z}, \hat{y}) \in P_N \cap \mathbb{B}^{m+n}$ para o PAGMGM, inicialmente é escolhido, por meio de um procedimento probabilístico, um conjunto de vértices $C \subset V$ de cardinalidade $|C|$ pré-determinada, que corresponde ao conjunto dos vértices centrais da solução $(\hat{z}, \hat{y})$, isto é, $\hat{y}_i = 0,\ \forall i \in C$ e $\hat{y}_i = 1,\ \forall i \in V \setminus C$. As regras empregadas para a definição do conjunto C serão apresentadas posteriormente. Para escolher o conjunto de arestas da solução $(\hat{z}, \hat{y})$, é inicialmente computada uma árvore geradora (idealmente de baixo custo) $T_C = (C, E_{T_C})$ do grafo $G_C = (C, E(C))$, valorado com custos $\{c_{ij} : \{i,j\} \in E(C)\}$. Assim, $\hat{z}_{ij} = 1,\ \forall \{i,j\} \in E_{T_C}$, $\hat{z}_{ij} = 0,\ \forall \{i,j\} \in E(C) \setminus E_{T_C}$ e $\hat{z}_{ij} = 0,\ \forall \{i,j\} \in E(V \setminus C)$. O motivo pelo qual definimos o conjunto de arestas baseado em uma árvore geradora de G_C, não necessariamente de custo mínimo, é tentar garantir que, no passo seguinte da heurística, quando são definidas as arestas que conectam as folhas aos vértices centrais, os graus dos vértices pertencentes ao conjunto C sejam maiores ou iguais a d e a solução represente uma árvore geradora viável para o PAGMGM.

Portanto, para obter a árvore geradora T_C, implementamos a seguinte adaptação do algoritmo de Kruskal para o PAGM. Assuma que em uma dada iteração do algoritmo adaptado de Kruskal, a aresta $\{p,q\} \in E(C)$ esteja sob avaliação para inclusão em E_{T_C}. Além de verificarmos se a seleção da aresta $\{p,q\}$ não forma um ciclo com as arestas já selecionadas, verificamos também se a desigualdade $\sum_{i \in C} \max\{0; d - \overline{g}_{T_C}(i)\} \leq n - |C|$ é satisfeita, em que $\overline{g}_{T_C}(i)$ denota o grau do vértice i na floresta em construção, após a inclusão da aresta $\{p,q\}$. Se essa condição adicional não for satisfeita, a aresta $\{p,q\}$ é descartada. Caso contrário, a aresta é incluída em E_{T_C}.

Sejam $\{g_{T_C}(i) : i \in C\}$ os graus (finais) dos vértices em T_C. Para definir o conjunto restante de arestas da solução viável para o PAGMGM, ou seja, o conjunto de arestas que conectam as folhas aos vértices centrais, resolvemos o Programa Linear:

$$\min \left\{ \sum_{\{i,j\} \in \delta(C)} c_{ij} \hat{z}_{ij} : \hat{z}_{\delta(C)} \in P_h \right\}, \tag{4.1}$$

em que o vetor $\hat{z}_{\delta(C)}$ corresponde às entradas $\hat{z}_{ij}$ de z tais que $\{i,j\} \in \delta(C)$ e o

poliedro $P_h \subset \mathbb{R}^{|\delta(C)|}$ é definido por:

$$\sum_{\{i,j\} \in \delta(C) \cap \delta(i)} \hat{z}_{ij} = 1, \ \ \forall i \in V \setminus C, \tag{4.2}$$

$$\sum_{\{i,j\} \in \delta(C) \cap \delta(i)} \hat{z}_{ij} \geq d - g_{T_C}(i), \ \ \forall i \in C, \tag{4.3}$$

$$\hat{z}_{ij} \geq 0, \ \ \forall \{i,j\} \in \delta(C). \tag{4.4}$$

As restrições (4.2) garantem que os graus dos vértices pertencentes a $V \setminus C$ são iguais a 1 (vértices folha). Por outro lado, as desigualdades (4.3) garantem que todo vértice em C tem grau maior ou igual a d em qualquer solução viável. Como o vetor de termos independentes é inteiro e o poliedro P_h é definido em termos de uma matriz de coeficientes totalmente unimodular [Wolsey, 1998], temos que P_h é um poliedro inteiro.

Isso posto, podemos afirmar que os passos (i) e (ii) da heurística CMP fornecem um vetor binário $(\hat{z}, \hat{y})$ que, por construção, representa um subgrafo conexo, acíclico e que satisfaz as restrições de grau mínimo, ou seja, uma solução viável para o PAGMGM. Cabe destacar que para instâncias definidas em grafos completos, a heurística CMP sempre encontra uma árvore geradora viável para o problema. No entanto, o procedimento não oferece garantia alguma de encontrar uma solução viável caso o problema seja definido em grafos esparsos.

Durante experimentos preliminares, observamos que as soluções ótimas do PAGMGM tendem a possuir conjuntos de vértices centrais de cardinalidade alta, bem próxima do limite superior estabelecido pela Proposição 4 e reescrito na restrição (3.23). Para as instâncias testadas neste estudo, definidas em grafos completos, percebemos que geralmente a folga na restrição (3.23) para as soluções ótimas é igual a zero ou bem pequena, comparada ao tamanho dessas instâncias. Na Figura 4.1, mostramos o número de ocorrências de cada possível valor das folgas em (3.23), para todas as 54 instâncias do PAGMGM resolvidas na otimalidade pelo algoritmo Branch-and-cut. Como pode ser visto, em apenas duas instâncias o valor da folga em (3.23) foi maior que 3. Portanto, na nossa implementação da heurística CMP, limitamos $|C|$ a assumir valores no intervalo $\left\lfloor \frac{n-2}{d-1} \right\rfloor - 3 \leq |C| \leq \left\lfloor \frac{n-2}{d-1} \right\rfloor$.

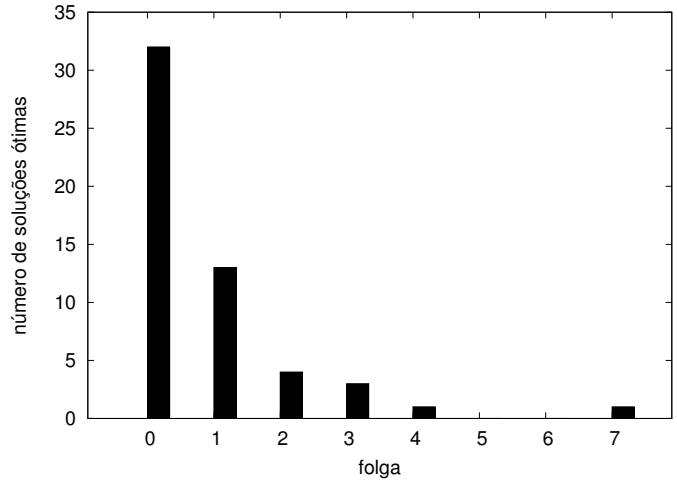

Figura 4.1. Número de soluções ótimas para cada um dos possíveis valores de folga da restrição (3.23).

Para concluir a descrição da heurística CMP, descrevemos a seguir a maneira como os conjuntos de vértices centrais C são escolhidos. Primeiramente, é calculada a árvore geradora mínima $T_V = (V, E_{T_V})$ de G, utilizando o algoritmo de Kruskal. Assuma que $\{i_1, i_2, \ldots, i_n\}$ são os vértices de V, ordenados em ordem não crescente de seus graus $\{g_{T_V}(j) : j \in V\}$ em T_V. Para todo $k = 1, \ldots, n$, é verificado se o vértice i_k é incluído no conjunto C, até que $|C|$ atinja o tamanho desejado. O vértice i_k é incluído no conjunto C de acordo com a probabilidade p_{i_k}, definida a seguir:

$$p_{i_k} = \begin{cases} 0{,}9 & \text{se} \quad g_{T_V}(i_k) \geq 2d, \\ 0{,}8 & \text{se} \quad d \leq g_{T_V}(i_k) < 2d, \\ 0{,}7 & \text{se} \quad d/2 \leq g_{T_V}(i_k) < d, \\ 0{,}6 & \text{se} \quad g_{T_V}(i_k) < d/2. \end{cases}$$

Essas probabilidades, que atribuem maiores chances de inclusão em C aos vértices com grau maior em T_V, foram definidas após a realização de experimentos preliminares.

Com o objetivo de aumentar as chances de encontrar uma boa solução viável para o PAGMGM, após a árvore geradora mínima T_V de G ser determinada, executamos várias vezes o procedimento probabilístico descrito, gerando vários conjuntos C de tamanhos desejados. Para cada possível tamanho do conjunto de vértices centrais C, executamos a heurística por um limite de tempo pré-determinado. Para cada conjunto C, uma árvore T_C correspondente é calculada e então um novo Programa Linear (4.1) é formulado e resolvido. A seguir, apresentamos o pseudo-código da

nossa heurística construtiva multipartida probabilística para o PAGMGM.

```
CMP(G = (V, E), d)
 1  n = |V|
 2  min = ⌊(n−2)/(d−1)⌋ − 3
 3  max = ⌈(n−2)/(d−1)⌉
 4  (z̄, ȳ) = NULL // melhor solução encontrada pela heurística
 5  cost(z̄, ȳ) = +∞
 6  central_nodes_sets = ∅ // pool de conjuntos de vértices centrais já considerados
 7  determine a árvore geradora mínima T_V de G
 8  ordene os vértices em ordem não crescente de seus graus em T_V, obtendo a lista {i_1, i_2, ..., i_n}
 9  for cardinality_C = min to max
10      time = 0
11      while time < time_limit // limite de tempo para cada cardinalidade de C
12          ŷ_i = 0,  ∀i ∈ V
13          C = ∅
14          k = 1;
15          while |C| ≠ cardinality_C and k ≠ n + 1
16              a = rand(0, 1) // distribuição uniforme no intervalo [0, 1]
17              if a ≤ p_{i_k}
18                  ŷ_{i_k} = 1
19                  C = C ∪ {v_{i_k}}
20              k = k + 1
21          if |C| ≠ cardinality_C or C ∈ central_nodes_sets
22              continue
23          central_nodes_sets = central_nodes_sets ∪ C
24          determine uma árvore geradora T_C = (C, E_{T_C}) de G_C = (C, E(C)),
            por meio do algoritmo adaptado de Kruskal
25          ẑ_ij = 1,  ∀{i, j} ∈ E_{T_C}
26          ẑ_ij = 0,  ∀{i, j} ∈ E(C) \ E_{T_C}
27          ẑ_ij = 0,  ∀{i, j} ∈ E(V \ C)
28          determine a solução ótima ẑ_{δ(C)} do Programa Linear (4.1)
29          if is_feasible(ẑ, ŷ) and Σ_{{i,j}∈E} c_ij ẑ_ij < cost(z̄, ȳ)
30              (z̄, ȳ) = (ẑ, ŷ)
31              cost(z̄, ȳ) = Σ_{{i,j}∈E} c_ij z̄_ij
32          atualize a variável time apropriadamente
33  return (z̄, ȳ)
```

Figura 4.2. Pseudo-código da Heurística Construtiva Multipartida Probabilística (CMP) para o PAGMGM

O algoritmo de Kruskal é executado uma única vez na heurística CMP (linha 7). Esse passo foi implementado com complexidade de tempo $O(m \log m)$. A ordenação dos vértices no passo seguinte é feita em $O(n \log n)$ (linha 8). Para cada

cardinalidade desejada do conjunto de vértices centrais C (linha 9), são geradas no laço principal da heurística várias soluções para o PAGMGM (linhas 11-32). Em cada iteração da heurística, determinamos o conjunto de vértices centrais C em $O(n)$ (linhas 12-20). Para evitar a determinação de árvores geradoras e a resolução do problema (4.1) para conjuntos repetidos de vértices centrais, armazenamos em uma tabela *hash* todos os conjuntos C gerados, com uma função *hash* de complexidade $O(n)$. Assim, os passos das linhas 21-23 são executados em $O(n)$. Em seguida, para determinar o conjunto de arestas da solução em construção, uma variação do algoritmo de Kruskal é executada para encontrar uma árvore geradora do subgrafo $G_C = (C, E(C))$ (linha 24), com complexidade de tempo $O(m \log n)$. As atribuições das linhas 25-27 são feitas em $O(m)$. Então, o Programa Linear (4.1) é resolvido (linha 28). Assumindo que q e v sejam, respectivamente, o número de restrições e variáveis do Programa Linear (4.1) em uma iteração da heurística CMP e que $f(q, v)$ seja a função de complexidade de tempo associada ao algoritmo utilizado para resolver (4.1), a seleção das arestas nas linhas 25-28 é feita em tempo proporcional a $O(m \log n + f(q, v))$. A avaliação de condições e as atualizações das linhas 29-32 são feitas em $O(m+n)$. Portanto, a complexidade de tempo para encontrar uma árvore geradora viável para o PAGMGM através da heurística proposta é $O(m \log m + f(q, v))$. Se a heurística CMP encontra uma ou mais soluções viáveis para o problema, ela retorna a melhor delas. Caso contrário, a heurística retorna $NULL$.

A heurística CMP foi utilizada para gerar uma solução inicial viável para os algoritmos Branch-and-cut e Local Branching propostos neste trabalho para a resolução exata do PAGMGM. Esses algoritmos são discutidos a seguir.

4.2 Um Algoritmo Branch-and-cut

O Branch-and-cut (BC) [Padberg & Rinaldi, 1991; Junger et al., 1995; Lucena & Beasley, 1996] é um método que incorpora geração de planos de corte a um procedimento enumerativo inteligente, do tipo Branch-and-bound, que avalia limites de Programação Linear na resolução dos subproblemas da árvore de enumeração. Na nossa implementação do método BC para a resolução do PAGMGM, empregamos o algoritmo de planos de corte que avalia o limite de PL da formulação P_D^r, apresentado na Subseção 3.5.1.1. A seguir, discutimos os detalhes de implementação do método.

4.2.1 Detalhes de Implementação

O algoritmo Branch-and-cut para o PAGMGM foi implementado através de chamadas às funções do resolvedor de Programação Inteira do CPLEX, versão 10.2, responsável por gerenciar a árvore de enumeração do BC. Todas as rotinas de pré-processamento, heurísticas e separação de desigualdades válidas (cortes baseados em arredondamento inteiro, cortes de Gomory, cortes da mochila, etc) disponibilizadas pelo CPLEX foram desativadas, e apenas as desigualdades descritas na Subseção 3.5.1.1 foram separadas, empregando o método de planos de corte introduzido para a avaliação de $w(P_D^r)$.

Com exceção do nó raiz da árvore de enumeração do BC, para o qual a separação das desigualdades DCUTs (3.11) é conduzida até que não sejam mais encontrados cortes violados, para todos os demais nós da árvore empregamos uma estratégia diferente. Considere que o método de planos de corte esteja sendo empregado para resolver o problema associado a um nó da árvore de enumeração diferente da raiz. Quando a desigualdade DCUT mais violada em uma iteração do método for violada por um valor pequeno, ou seja, quando a capacidade do corte de capacidade mínima do grafo $\overline{D}$ for próxima de 1, nenhuma desigualdade DCUT é incluída na formulação para reotimização. Portanto, na iteração em que nenhuma DCUT é inserida na formulação e todas as restrições lógicas (3.21) são satisfeitas, o método de planos de corte é encerrado para a resolução daquele nó.

Mais precisamente, considere que $(\overline{x}^r, \overline{y})$ denote a solução do problema relaxado associado a um nó da árvore de enumeração. Sabemos que se $\delta^-(W)$ é um corte em $\overline{D}$ de capacidade $\sum_{(p,q)\in\delta^-(W)} \overline{x}^r_{pq}$ menor que 1, então a desigualdade $x^r(\delta^-(W)) \geq 1$ é violada por $(\overline{x}^r, \overline{y})$ e pode ser inserida na formulação relaxada para reotimização. No entanto, empregamos uma outra estratégia em que apenas desigualdades associadas a cortes de capacidade inferiores a um limiar $\kappa \in (0, 1)$ são inseridas na formulação. Experimentos computacionais realizados mostraram que o método BC obteve um melhor desempenho quando utilizamos $\kappa = 0{,}85$. Assim, este valor foi utilizado em todos os experimentos que empregaram o BC neste trabalho.

Ao final do algoritmo de planos de corte empregado em cada nó, toda vez que a solução da relaxação de Programação Linear for fracionária e seu custo for menor que o custo da melhor solução viável encontrada para o PAGMGM, a subdivisão do espaço de busca na árvore de enumeração é realizada, com maior prioridade associada às ramificações nas variáveis y, de acordo com a opção *strong branching* [Applegate et al., 1995] do CPLEX. Todos os outros parâmetros *default* do CPLEX

foram mantidos.

Para cada instância, estabelecemos um limite de tempo máximo de 21600 segundos (6 horas) para a execução do BC. Para as instâncias da classe *ALM*, executamos a heurística CMP durante 300 segundos, 75 segundos para cada valor possível de $|C|$ ($\lfloor \frac{n-2}{d-1} \rfloor - 3 \leq |C| \leq \lfloor \frac{n-2}{d-1} \rfloor$), e utilizamos a melhor solução viável encontrada como solução primal para o algoritmo BC. Para as instâncias das classes *CRD* e *SYM*, optamos por não utilizar a heurística, uma vez que o método BC é capaz de encontrar, para os tamanhos dos problemas dessas classes, soluções viáveis de melhor qualidade logo no início da enumeração. Descrevemos a seguir os resultados computacionais obtidos pelo método.

4.2.2 Resultados Computacionais

Nesta seção, avaliamos computacionalmente o algoritmo BC proposto para o PAGMGM, e comparamos os seus resultados àqueles obtidos pelos melhores métodos da literatura já empregados para a resolução do problema, a saber: as meta-heurísticas propostas em [Martins & Souza, 2009] e o algoritmo Branch-and-bound proposto em [Akgún & Tansel, 2010].

Todas as 90 instâncias de teste consideradas neste estudo foram utilizadas nos experimentos relatados nesta seção. Para compararmos o BC com os melhores métodos propostos na literatura para o PAGMGM, implementamos o algoritmo Branch-and-bound baseado na formulação MTZ (BB-MTZ), proposto em [Akgún & Tansel, 2010], utilizando o pacote de otimização CPLEX 10.2, com todos os seus parâmetros *default*. Nos experimentos realizados, utilizamos um limite de tempo de 6 horas para a execução do BB-MTZ, da mesma forma como fizemos com o BC, e avaliamos os dois métodos em um mesmo ambiente computacional, descrito na Subseção 3.6.2. Por outro lado, para a comparação do BC com as meta-heurísticas propostas em [Martins & Souza, 2009] (HVNS), utilizamos os resultados obtidos nos experimentos computacionais relatados naquele estudo, realizados em um ambiente computacional diferente do nosso, em uma máquina com processador Pentium IV, $3{,}2$ GHz, com 512 MB de memória RAM.

Dessa forma, torna-se difícil comparar os tempos computacionais gastos pelas meta-heurísticas HVNS e pelo algoritmo BC, uma vez que seus experimentos computacionais foram conduzidos em máquinas com configurações diferentes. No entanto, com o intuito de apresentar ao leitor os limites superiores conhecidos na literatura para o PAGMGM, obtidos pelas meta-heurísticas HVNS, apresentamos nesta seção os resultados computacionais relatados em [Martins & Souza, 2009]. Cabe

ressaltar que, como diversas heurísticas foram propostas para o PAGMGM por Martins & Souza [2009], decidimos considerar *apenas o melhor* resultado apresentado naquele estudo para cada instância. Desse modo, daqui em diante no texto, ao mencionarmos o *método HVNS*, estamos nos referindo a todas as meta-heurísticas propostas em [Martins & Souza, 2009], sempre considerando o melhor resultado (em termos de limite superior) apresentado para cada instância naquele estudo.

Para compararmos os métodos HVNS, BB-MTZ e BC, apresentamos primeiramente uma síntese dos resultados obtidos por esses métodos, para em seguida descrevermos os resultados de maneira detalhada. A Tabela 4.1 sumariza os resultados obtidos pelos três métodos para as 90 instâncias de teste avaliadas. Para cada tamanho de instância considerado, uma linha da tabela mostra os resultados obtidos por cada método. As duas primeiras colunas da tabela indicam o valor de n e o número de instâncias testadas, definidas em grafos com n vértices. A coluna seguinte apresenta as razões entre os melhores limites superiores fornecidos pelos métodos BC e HVNS $\left(\frac{BC(\overline{w})}{HVNS(\overline{w})}\right)$. Para essa comparação, estabelecemos um limite de tempo de execução para o BC igual ao utilizado em [Martins & Souza, 2009], que depende do tamanho da instância (veja a coluna lt das Tabelas 4.2 e 4.3 para conhecer esses limites de tempo). Nas três colunas seguintes da tabela, apresentamos, respectivamente, o número de instâncias resolvidas pelo BB-MTZ (resol.), isto é, o número de instâncias para as quais o BB-MTZ foi capaz de encontrar a solução ótima e prover um certificado para sua otimalidade, a soma do tempo total gasto, em segundos, para o BB-MTZ resolver essas instâncias (tt) e o *gap* percentual médio entre os melhores limites inferiores e superiores obtidos pelo BB-MTZ para as instâncias não resolvidas. As últimas três colunas da tabela apresentam esses mesmos tipos de dados (resol., tt e *gap*) associados aos resultados obtidos pelo BC.

Cabe destacar que os valores indicados na coluna tt associada ao método BC, correspondem ao somatório dos tempos gastos pelo BC para resolver as instâncias que também foram resolvidas na otimalidade pelo BB-MTZ, para cada tamanho de grafo considerado. Para os valores de n em que os métodos BB-MTZ ou BC não foram capazes de resolver nenhuma instância, o símbolo "-" é apresentado nas entradas correspondentes das colunas tt da tabela. De maneira análoga, para os valores de n em que os algoritmos BB-MTZ ou BC resolveram todas as instâncias, é apresentado o símbolo "-" nas entradas correspondentes das colunas *gap* (%), indicando que esses valores foram iguais a zero.

Os resultados computacionais da Tabela 4.1 mostram que a diferença entre os custos das melhores soluções viáveis encontradas pelos métodos HVNS e BC aumenta à medida que instâncias maiores são consideradas. Observe que para as

Tabela 4.1. Síntese dos resultados obtidos pelos métodos HVNS [Martins & Souza, 2009], BB-MTZ [Akgún & Tansel, 2010] e BC, para as instâncias das classes *CRD*, *SYM* e *ALM*.

		HVNS x BC	BB-MTZ			BC		
n	#instâncias	$\frac{BC(\overline{w})}{HVNS(\overline{w})}$	resol.	tt	*gap* (%)	resol.	tt	*gap* (%)
30	12	1,0000	12	357,3	-	12	5,3	-
50	18	0,9991	18	35757,0	-	18	521,3	-
70	18	0,9879	12	1812,5	4,50	18	454,4	-
100	12	0,9913	6	55977,6	8,68	6	7242,7	4,10
200	6	0,9414	0	-	10,01	0	-	7,24
300	6	0,9256	0	-	14,85	0	-	14,07
400	9	0,9120	0	-	17,30	0	-	13,55
500	9	0,9131	0	-	32,56	0	-	16,13

60 instâncias com até 100 vértices, das quais o BC conseguiu resolver 54, o método HVNS encontrou soluções com custos próximos aos obtidos pelo BC. Por outro lado, para as 30 instâncias com 200 ou mais vértices, as soluções fornecidas pelo BC foram em média $8{,}54\%$ melhores que as obtidas pelas meta-heurísticas HVNS.

Em uma comparação entre os métodos BC e BB-MTZ, primeiramente destacamos que o BC resolveu todas as 18 instâncias definidas em grafos com 70 vértices, enquanto o BB-MTZ não foi capaz de resolver 6 delas, obtendo um *gap* de dualidade médio igual a $4{,}5\%$ nesses casos. A tabela também indica que o tempo computacional gasto pelo BC para resolver as instâncias de 30 e 50 vértices foi duas ordens de grandeza inferior ao tempo correspondente gasto pelo BB-MTZ. Para as instâncias de 70 e 100 vértices resolvidas pelos dois métodos, o BC gastou um tempo inferior a 15% do tempo gasto pelo BB-MTZ. Além disso, note que os *gaps* percentuais médios do BC foram inferiores aos obtidos pelo BB-MTZ em todos os casos, com uma diferença considerável para as instâncias de 500 vértices, em que o *gap* de dualidade médio do BC foi aproximadamente 50% inferior ao *gap* do BB-MTZ.

Ainda assim, os resultados mostrados na Tabela 4.1 indicam claramente que o BC teve dificuldades para tratar as maiores instâncias testadas. Como pode ser observado, o método não foi capaz de resolver nenhuma instância com 200 ou mais vértices, e nesses casos, o *gap* de dualidade observado foi consideravelmente alto, principalmente para as instâncias com 300 ou mais vértices.

4.2.2.1 Resultados Detalhados

Nesta seção, apresentamos os resultados computacionais detalhados dos experimentos realizados com os métodos HVNS, BB-MTZ e BC.

As Tabelas 4.2 e 4.3 apresentam os resultados obtidos pelos métodos HVNS

e BC. A primeira delas mostra os resultados dos métodos para as instâncias das classes *CRD* e *SYM*, enquanto que a segunda apresenta os resultados obtidos pelos métodos para as instâncias da classe *ALM*. Esse mesmo padrão de divisão das tabelas para a apresentação dos resultados (uma tabela com os resultados para as classes *CRD* e *SYM* e outra tabela para a classe *ALM*) é utilizado nas tabelas seguintes desta dissertação.

A primeira coluna da Tabela 4.2 indica os ids das instâncias consideradas. Nas duas colunas seguintes são indicados os resultados obtidos pelas meta-heurísticas HVNS: o melhor limite superior obtido ($\overline{w}$) e o limite de tempo total imposto para a execução do método (tl, em segundos). As cinco colunas seguintes apresentam os resultados obtidos pelo BC, associados à obtenção de limites superiores: o melhor limite superior obtido no mesmo limite de tempo estabelecido para o HVNS ($\overline{w}_{lt}$), o tempo computacional necessário para obtê-lo ($t_{\overline{w}_{lt}}$, em segundos), o melhor limite superior obtido pelo BC no limite de tempo de 6 horas ($\overline{w}$), o tempo computacional necessário para obtê-lo ($t_{\overline{w}}$, em segundos) e o tempo total durante o qual o BC foi executado (t, em segundos). As colunas da Tabela 4.3 são similares às colunas da Tabela 4.2. A única diferença é uma coluna adicional para indicar o melhor limite superior obtido pela heurística CMP após 300 segundos de execução ($CMP(\overline{w})$), utilizado como limite primal inicial do BC para as instâncias da classe *ALM*.

Os resultados computacionais indicam que, em praticamente todos os casos, os melhores limites superiores obtidos pelo BC foram pelo menos tão bons quanto os fornecidos pelas heurísticas HVNS. Considerando o mesmo limite de tempo para os dois métodos, em 61 casos os limites primais obtidos pelo BC foram mais fortes que os dados pelo método HVNS. Em apenas dois casos (instâncias com ids 57 e 89), o melhor limite superior dado pelas heurísticas HVNS foi melhor que o fornecido pelo BC.

Como pode ser observado na Tabela 4.3, o BC não foi capaz de encontrar uma solução viável melhor que a solução inicial fornecida pela heurística CMP para uma instância de 400 vértices (id 81) e duas instâncias de 500 vértices (ids 88 e 89), mesmo após quase 6 horas de processamento. No entanto, os limites primais fornecidos pela heurística CMP foram de qualidade comparável àqueles fornecidos pelo método HVNS. De fato, em dois desses três casos, a solução dada pela heurística CMP foi melhor que a encontrada pelas heurísticas HVNS.

As Tabelas 4.4 e 4.5 apresentam resultados computacionais detalhados obtidos pelos algoritmos BB-MTZ e BC. A primeira coluna das tabelas indica os ids das instâncias consideradas. As cinco colunas seguintes apresentam os resultados obtidos pelo método BB-MTZ: os melhores limites inferior ($\underline{w}$) e superior ($\overline{w}$) obtidos no

limite de tempo de 6 horas, o *gap* de dualidade em valores percentuais, dado por $100(\overline{w} - \underline{w})/\overline{w}$, o tempo total de execução do algoritmo (t, em segundos) e o número total de nós investigados na árvore de enumeração (nós). Entradas similares são apresentadas para o algoritmo BC nas cinco colunas subsequentes. Quando uma instância é resolvida na otimalidade, a entrada correspondente nas colunas *gap* (%) mostram o símbolo "-".

Como pode ser observado nas tabelas, o método BC claramente domina o BB-MTZ em termos de tempos computacionais. Para algumas instâncias, o tempo gasto pelo BC para prover o certificado de otimalidade foi três ou quatro ordens de grandeza menor que o tempo correspondente gasto pelo BB-MTZ. Para a instância com id 7, por exemplo, enquanto o BB-MTZ gastou $20283{,}90$ segundos para resolvê-la, o método BC o fez em $3{,}93$ segundos. Além disso, todas as instâncias resolvidas na otimalidade pelos dois métodos foram resolvidas mais rapidamente pelo BC. No total, o BC gastou $8{,}76\%$ do tempo gasto pelo BB-MTZ para resolvê-las.

Como consequência direta dos fracos limites de PL fornecidos pela formulação MTZ (veja Tabelas 3.2 e 3.3), o método BB-MTZ geralmente precisa investigar muito mais nós na árvore de enumeração que o algoritmo BC. Para as instâncias resolvidas por ambos os métodos, o número de nós investigados no total pelo BC foi $0{,}35\%$ do número total de nós investigados pelo BB-MTZ. Apesar do tempo gasto para investigar um nó da árvore de enumeração do BB-MTZ ser, em geral, menor que o tempo correspondente gasto na árvore de enumeração do BC, essa diferença elevada no número de nós investigados por cada método implica diretamente nas diferenças percebidas entre os seus tempos computacionais.

Em relação aos melhores limites inferiores obtidos pelos dois métodos, em 34 das 36 instâncias não resolvidas por nenhum dos algoritmos, o limite inferior fornecido pelo BC foi mais forte que o dado pelo BB-MTZ. Considerando essas 36 instâncias, em média, o melhor limite inferior dado pelo BC foi $2{,}16\%$ mais forte que o fornecido pelo BB-MTZ. Pelo lado primal, percebemos um equilíbrio entre os dois métodos em termos do número de melhores limites superiores obtidos. Considerando as 36 instâncias não resolvidas, o BC forneceu 19 melhores limites superiores enquanto que o BB-MTZ forneceu 16. Em apenas uma dessas instâncias (id 57), os limites superiores obtidos pelos dois métodos foram iguais. Na média, para as instâncias não resolvidas, os melhores limites superiores dados pelo BC foram $5{,}26\%$ mais fortes que aqueles dados pelo método BB-MTZ. Esse elevado valor é devido principalmente à dificuldade do método BB-MTZ em obter boas soluções viáveis para o PAGMGM para algumas instâncias de 500 vértices (veja os custos das melhores soluções encontradas pelo BB-MTZ para as instância com ids 86, 88 e 90).

Para as 12 instâncias de 100 vértices (ids 25 a 30 e 55 a 60), os métodos BB-MTZ e BC foram capazes de resolver as mesmas 6 instâncias. Note que essas instâncias resolvidas pelos dois métodos foram definidas com o parâmetro de grau mínimo $d = 10$, enquanto que as instâncias em aberto, isto é, cuja otimalidade não foi provada, possuem parâmetro de grau mínimo $d = 5$. Resultados similares podem ser observados para instâncias de outros tamanhos de grafos, indicando que quanto menor o valor de d, mais difícil é a resolução do PAGMGM por meio de algoritmos de enumeração como o BB-MTZ e o BC. De fato, para todos os valores de n (exceto 500) avaliados neste estudo, o *gap* de dualidade médio obtido pelo BC diminui à medida que o valor de d aumenta.

Cabe ressaltar que os resultados apresentados neste trabalho obtidos com o algoritmo BB-MTZ aprimoram os resultados relatados em [Akgún & Tansel, 2010]. Uma vez que o método que implementamos corresponde exatamente ao algoritmo BB baseado na mais forte das formulações MTZ apresentadas naquele estudo, as razões para essa melhora nos resultados são relacionadas exclusivamente ao ambiente de execução, versão do CPLEX, e limite de tempo utilizados, assim como às instâncias testadas, já que apenas instâncias com até 50 vértices haviam sido avaliadas em [Akgún & Tansel, 2010]. Assim, todas as análises descritas neste texto relacionadas ao algoritmo BB-MTZ e aos melhores resultados obtidos (na literatura) por este método, são baseadas nos experimentos computacionais descritos nesta dissertação.

Isso posto, podemos destacar que o algoritmo BC proposto nesta seção forneceu 6 novos certificados de otimalidade para instâncias do PAGMGM, assim como 34 novos melhores limites inferiores e 22 novos melhores limites superiores, considerando as 36 instâncias em aberto. Apesar desses resultados obtidos pelo BC, o problema ainda é muito difícil de ser resolvido, principalmente para as maiores instâncias da classe *ALM*. Com o intuito de tentar obter melhores soluções viáveis para o PAGMGM em menores tempos computacionais, implementamos um algoritmo Local Branching, que emprega o próprio BC como resolvedor interno. Discutimos tal método a seguir.

Tabela 4.2. Comparação entre as meta-heurísticas HVNS propostas em [Martins & Souza, 2009] e o algoritmo BC, para as instâncias das classes *CRD* e *SYM*.

	HVNS		BC				
id	$\overline{w}$	lt	$\overline{w}_{lt}$	$t_{\overline{w}_{lt}}$	$\overline{w}$	$t_{\overline{w}}$	t
1	4026	900	4026	0,2	4026	0,2	0,2
2	3793	900	3793	0,1	3793	0,1	0,1
3	4293	900	4293	0,2	4293	0,2	0,2
4	5026	900	5026	0,7	5026	0,7	0,7
5	4648	900	4648	0,2	4648	0,2	0,2
6	5425	900	5425	1,7	5425	1,7	2,5
7	5512	1800	5512	3,3	5512	3,3	3,9
8	5813	1800	5813	49,8	5813	49,8	50,2
9	5590	1800	5590	2,4	5590	2,4	3,0
10	6908	1800	6908	70,0	6908	70,0	70,6
11	7238	1800	7204	68,3	7204	68,3	77,7
12	7277	1800	7277	269,1	7277	269,1	274,3
13	9633	1800	9633	8,3	9633	8,3	8,8
14	9743	1800	9743	3,8	9743	3,8	5,7
15	9855	1800	9855	4,5	9855	4,5	5,1
16	6609	3600	6516	7,2	6516	7,2	118,8
17	6621	3600	6586	2288,3	6586	2288,3	2299,5
18	7058	3600	7053	2062,0	7053	2062,0	2065,6
19	8177	3600	8140	972,6	8134	15309,0	15316,9
20	7971	3600	7943	1139,9	7943	1139,9	1143,6
21	8419	3600	8419	1049,1	8419	1049,1	1198,4
22	11355	3600	11235	62,1	11235	62,1	66,1
23	11395	3600	11373	47,0	11373	47,0	180,2
24	11986	3600	11979	16,1	11979	16,1	18,2
25	9387	7200	9382	4711,2	9349	20177,9	21600,0
26	9728	7200	9572	1674,3	9571	9333,1	21600,0
27	9739	7200	9676	6573,3	9653	16813,7	21600,0
28	13006	7200	12916	1051,1	12916	1051,1	1622,7
29	13255	7200	13026	443,7	13026	443,7	445,0
30	13365	7200	13365	322,0	13365	322,0	577,5
31	1197	900	1197	0,1	1197	0,1	0,1
32	1435	900	1435	0,1	1435	0,1	0,1
33	1408	900	1408	0,1	1408	0,1	0,1
34	1765	900	1765	0,7	1765	0,7	0,7
35	2090	900	2090	0,2	2090	0,2	0,2
36	2008	900	2008	0,1	2008	0,1	0,1
37	1298	1800	1278	0,5	1278	0,5	0,5
38	1188	1800	1178	0,5	1178	0,5	0,5
39	1620	1800	1615	0,2	1615	0,2	0,2
40	2054	1800	2054	5,1	2054	5,1	5,2
41	1760	1800	1760	0,9	1760	0,9	0,9
42	2541	1800	2525	6,4	2525	6,4	7,0
43	4121	1800	4121	2,5	4121	2,5	2,5
44	4166	1800	4166	1,0	4166	1,0	1,5
45	4979	1800	4979	3,0	4979	3,0	3,8
46	1362	3600	1360	1,3	1360	1,3	1,3
47	1471	3600	1448	3,9	1448	3,9	4,1
48	1551	3600	1521	4,3	1521	4,3	4,4
49	2240	3600	2028	6,9	2028	6,9	8,9
50	2496	3600	2165	1,9	2165	1,9	1,9
51	2242	3600	2210	19,4	2210	19,4	19,4
52	5055	3600	4979	62,8	4979	62,8	63,5
53	4912	3600	4787	29,4	4787	29,4	36,1
54	5098	3600	4997	23,5	4997	23,5	50,4

Tabela 4.3. Comparação entre as meta-heurísticas HVNS propostas em [Martins & Souza, 2009] e o algoritmo BC, para as instâncias da classe *ALM*.

	HVNS		BC					
id	$\overline{w}$	lt	$CPM(\overline{w})$	$\overline{w}_{lt}$	$t_{\overline{w}_{lt}}$	$\overline{w}$	$t_{\overline{w}}$	t
55	5439	7200	5575	5353	6408,0	5353	6408,0	21600,0
56	5207	7200	5532	5057	4369,3	5055	7731,7	21600,0
57	5456	7200	5893	5486	2712,8	5450	20656,4	21600,0
58	7180	7200	7521	7164	130,2	7164	130,2	158,3
59	6915	7200	7280	6886	853,3	6886	853,3	857,8
60	7509	7200	7690	7382	1556,2	7382	1556,2	3581,3
61	7467	10800	7828	7125	6613,8	7075	16366,6	21600,0
62	7680	10800	8239	7258	4136,1	7245	19397,2	21600,0
63	8217	10800	8452	7492	1222,2	7492	1222,2	21600,0
64	10391	10800	10404	9656	9812,7	9652	12749,4	21600,0
65	10238	10800	10765	9820	9850,8	9820	9850,8	21600,0
66	10533	10800	11370	9978	8539,7	9978	8539,7	21600,0
67	9871	14400	10511	9705	1748,8	9705	1748,8	21600,0
68	10532	14400	10051	9407	1522,8	9407	1522,8	21600,0
69	10887	14400	10984	9458	3133,5	9458	3133,5	21600,0
70	13899	14400	13439	12891	3417,1	12891	3417,1	21600,0
71	13210	14400	13570	12246	1935,5	12246	1935,5	21600,0
72	13792	14400	14035	13110	2144,1	13110	2144,1	21600,0
73	12487	18000	12124	11079	5241,3	11079	5241,3	21600,0
74	13877	18000	12155	11036	4670,7	11036	4670,7	21600,0
75	12379	18000	12524	10711	10746,7	10711	10746,7	21600,0
76	17309	18000	17029	14611	12527,7	14611	12527,7	21600,0
77	16595	18000	16578	15086	5585,1	15086	5585,1	21600,0
78	16439	18000	15582	14506	5895,1	14506	5895,1	21600,0
79	21339	18000	22905	21260	14805,9	21260	14805,9	21600,0
80	21299	18000	22772	20473	12807,5	20473	12807,5	21600,0
81	22049	18000	21478	21478	300,0	21478	300,0	21600,0
82	14626	21600	13328	11881	15773,7	11881	15773,7	21600,0
83	14039	21600	12990	12518	14871,7	12518	14871,7	21600,0
84	13521	21600	12997	12035	12348,7	12035	12348,7	21600,0
85	19342	21600	17652	15617	20011,0	15617	20011,0	21600,0
86	18138	21600	17891	16344	13718,6	16344	13718,6	21600,0
87	18269	21600	18644	16466	15514,9	16466	15514,9	21600,0
88	24999	21600	24508	24508	300,0	24508	300,0	21600,0
89	24823	21600	24900	24900	300,0	24900	300,0	21600,0
90	25468	21600	25064	23901	20303,5	23901	20303,5	21600,0

Tabela 4.4. Comparação entre o algoritmo BB-MTZ proposto em [Akgún & Tansel, 2010] e o algoritmo BC, para as instâncias das classes *CRD* e *SYM*.

	BB-MTZ					BC				
id	$\underline{w}$	$\overline{w}$	*gap* (%)	t	nós	$\underline{w}$	$\overline{w}$	*gap* (%)	t	nós
1	4026,0	4026	-	5,5	4223	4026,0	4026	-	0,2	37
2	3793,0	3793	-	329,4	308419	3793,0	3793	-	0,1	52
3	4293,0	4293	-	2,1	1304	4293,0	4293	-	0,2	55
4	5026,0	5026	-	3,3	547	5026,0	5026	-	0,7	54
5	4648,0	4648	-	5,8	2826	4648,0	4648	-	0,2	20
6	5425,0	5425	-	7,5	2739	5425,0	5425	-	2,5	149
7	5512,0	5512	-	20283,9	5691339	5512,0	5512	-	3,9	471
8	5813,0	5813	-	6920,1	1801405	5813,0	5813	-	50,2	4233
9	5590,0	5590	-	602,7	162192	5590,0	5590	-	3,0	282
10	6908,0	6908	-	3150,4	522125	6908,0	6908	-	70,6	2077
11	7204,0	7204	-	944,4	103530	7204,0	7204	-	77,7	1418
12	7277,0	7277	-	3713,2	619438	7277,0	7277	-	274,3	5080
13	9633,0	9633	-	23,3	1161	9633,0	9633	-	8,8	131
14	9743,0	9743	-	16,7	697	9743,0	9743	-	5,7	78
15	9855,0	9855	-	18,1	861	9855,0	9855	-	5,1	65
16	6267,4	6516	3,82	21600,0	1313337	6516,0	6516	-	118,8	6893
17	6211,9	6621	6,18	21600,0	1191255	6586,0	6586	-	2299,5	89134
18	6569,7	7066	7,02	21600,0	1145801	7053,0	7053	-	2065,6	78291
19	7699,3	8134	5,34	21600,0	1057801	8134,0	8134	-	15316,9	121735
20	7726,5	7943	2,73	21600,0	1029342	7943,0	7943	-	1143,6	11366
21	8260,3	8421	1,91	21600,0	988001	8419,0	8419	-	1198,4	12039
22	11235,0	11235	-	143,4	5052	11235,0	11235	-	66,1	401
23	11373,0	11373	-	632,9	28075	11373,0	11373	-	180,2	1169
24	11979,0	11979	-	69,7	1778	11979,0	11979	-	18,2	142
25	8559,2	9334	8,30	21600,0	495575	8839,3	9349	5,45	21600,0	51095
26	8964,1	9562	6,25	21600,0	450623	9303,6	9571	2,79	21600,0	59482
27	8699,4	9635	9,71	21600,0	529901	9228,3	9653	4,40	21600,0	62227
28	12916,0	12916	-	18939,7	458465	12916,0	12916	-	1622,7	3868
29	13026,0	13026	-	833,9	10761	13026,0	13026	-	445,0	1841
30	13365,0	13365	-	7974,3	151036	13365,0	13365	-	577,5	2333
31	1197,0	1197	-	0,2	3	1197,0	1197	-	0,1	8
32	1435,0	1435	-	0,2	0	1435,0	1435	-	0,1	7
33	1408,0	1408	-	0,1	0	1408,0	1408	-	0,0	4
34	1765,0	1765	-	1,6	68	1765,0	1765	-	0,7	34
35	2090,0	2090	-	1,1	36	2090,0	2090	-	0,2	14
36	2008,0	2008	-	0,6	0	2008,0	2008	-	0,1	6
37	1278,0	1278	-	2,1	5	1278,0	1278	-	0,5	24
38	1178,0	1178	-	0,9	3	1178,0	1178	-	0,5	20
39	1615,0	1615	-	1,0	2	1615,0	1615	-	0,2	7
40	2054,0	2054	-	10,5	444	2054,0	2054	-	5,2	135
41	1760,0	1760	-	3,6	42	1760,0	1760	-	0,9	21
42	2525,0	2525	-	20,2	1639	2525,0	2525	-	7,0	166
43	4121,0	4121	-	7,8	160	4121,0	4121	-	2,5	30
44	4166,0	4166	-	14,1	566	4166,0	4166	-	1,5	20
45	4979,0	4979	-	24,1	1173	4979,0	4979	-	3,8	47
46	1360,0	1360	-	3,6	34	1360,0	1360	-	1,3	36
47	1448,0	1448	-	9,2	232	1448,0	1448	-	4,1	138
48	1521,0	1521	-	24,2	1306	1521,0	1521	-	4,4	107
49	2028,0	2028	-	25,2	347	2028,0	2028	-	8,9	89
50	2165,0	2165	-	34,0	743	2165,0	2165	-	1,9	27
51	2210,0	2210	-	47,4	1110	2210,0	2210	-	19,4	210
52	4979,0	4979	-	163,9	3385	4979,0	4979	-	63,5	369
53	4787,0	4787	-	238,0	6735	4787,0	4787	-	36,1	190
54	4997,0	4997	-	421,0	11827	4997,0	4997	-	50,4	298

Tabela 4.5. Comparação entre o algoritmo BB-MTZ proposto em [Akgún & Tansel, 2010] e o algoritmo BC, para as instâncias da classe *ALM*.

	BB-MTZ					BC				
id	$\underline{w}$	$\overline{w}$	*gap* (%)	t	nós	$\underline{w}$	$\overline{w}$	*gap* (%)	t	nós
55	4831,2	5457	11,47	21600,0	276124	5142,6	5353	3,93	21600,0	47840
56	4719,4	5197	9,19	21600,0	198941	4870,3	5055	3,65	21600,0	44520
57	5058,5	5450	7,18	21600,0	213601	5210,2	5450	4,40	21600,0	42519
58	7164,0	7164	-	2269,8	20300	7164,0	7164	-	158,3	544
59	6886,0	6886	-	5712,4	32585	6886,0	6886	-	857,8	2619
60	7382,0	7382	-	20247,4	153172	7382,0	7382	-	3581,3	6464
61	6183,4	7217	14,32	21600,0	41301	6429,0	7075	9,13	21600,0	10235
62	6361,2	7180	11,40	21600,0	42701	6597,9	7245	8,93	21600,0	12332
63	6626,4	7556	12,30	21600,0	38359	6882,5	7492	8,13	21600,0	9514
64	8919,8	9680	7,85	21600,0	23573	9017,9	9652	6,57	21600,0	5572
65	9263,0	9916	6,59	21600,0	25235	9350,7	9820	4,78	21600,0	6713
66	9278,3	10039	7,58	21600,0	22738	9391,2	9978	5,88	21600,0	4810
67	7780,2	9211	15,53	21600,0	13905	8016,0	9705	17,40	21600,0	3744
68	7564,5	9295	18,62	21600,0	13885	7798,8	9407	17,10	21600,0	3433
69	7870,4	9359	15,91	21600,0	12662	8099,0	9458	14,37	21600,0	3594
70	11280,5	12603	10,49	21600,0	4192	11355,4	12891	11,91	21600,0	1862
71	10826,1	12723	14,91	21600,0	3311	10936,2	12246	10,70	21600,0	2093
72	11279,2	13063	13,66	21600,0	3002	11416,1	13110	12,92	21600,0	1742
73	8837,9	10917	19,04	21600,0	9267	9121,2	11079	17,67	21600,0	1568
74	8967,4	10791	16,90	21600,0	8475	9227,9	11036	16,38	21600,0	1626
75	8630,6	10336	16,50	21600,0	17481	8945,5	10711	16,48	21600,0	1532
76	12927,9	15055	14,13	21600,0	3307	12976,0	14611	11,19	21600,0	834
77	12779,1	14774	13,50	21600,0	3071	12876,1	15086	14,65	21600,0	904
78	12403,5	15142	18,09	21600,0	2346	12601,9	14506	13,13	21600,0	1005
79	18932,9	23287	18,70	21600,0	251	18929,0	21260	10,96	21600,0	230
80	18675,7	23951	22,03	21600,0	252	18712,8	20473	8,60	21600,0	454
81	18667,5	22445	16,83	21600,0	272	18708,4	21478	12,89	21600,0	174
82	9608,3	11872	19,07	21600,0	6401	9868,9	11881	16,94	21600,0	626
83	9756,4	12512	22,02	21600,0	9301	10077,0	12518	19,50	21600,0	676
84	9543,0	12190	21,71	21600,0	7648	9904,7	12035	17,70	21600,0	734
85	13627,0	17900	23,87	21600,0	310	13736,4	15617	12,04	21600,0	333
86	14131,0	33992	58,43	21600,0	246	14258,5	16344	12,76	21600,0	378
87	13972,8	17168	18,61	21600,0	824	14071,7	16466	14,54	21600,0	410
88	19839,6	42194	52,98	21600,0	183	19851,2	24508	19,00	21600,0	140
89	20396,4	23871	14,56	21600,0	144	20419,4	24900	17,99	21600,0	129
90	20395,8	53380	61,79	21600,0	193	20384,0	23901	14,71	21600,0	167

4.3 Um Algoritmo Local Branching

O Local Branching (LB) é um algoritmo exato proposto por Fischetti & Lodi [2003] para resolver problemas de Programação Inteira. O método pode ser interpretado como uma formalização de procedimentos de busca local através de modelos de Programação Matemática. Mais precisamente, o LB define e controla subespaços de soluções viáveis por meio de um arcabouço externo de ramificação, e as explora parcial ou totalmente através de um resolvedor genérico de Programação Inteira.

O LB segue o espírito das meta-heurísticas de busca local. Entretanto, as vizinhanças de uma dada solução são determinadas através da introdução de desigualdades lineares ao modelo de Programação Inteira empregado para formular o problema original, definindo um novo subproblema de PI. Assim, uma das vantagens em se utilizar o método é poder usufruir dos enormes esforços dedicados atualmente às pesquisas e implementações dos resolvedores de PI, e utilizá-los como *caixa-preta* para resolver cada subproblema definido pelo arcabouço externo de ramificação do método LB.

A ideia do LB consiste em resolver, desde os estágios iniciais do processamento, subproblemas associados a regiões menores e mais "promissoras" do espaço de busca de soluções, definidas a partir da vizinhança de uma solução conhecida para o problema. Na prática, observa-se que o método é capaz de encontrar de forma mais rápida soluções viáveis de boa qualidade [Fischetti & Lodi, 2003]. O método LB vem sendo empregado para resolver diversos problemas de Otimização Combinatória, incluindo problemas em áreas como Roteamento de Veículos [Rei et al., 2010; Valle et al., 2011] e Projeto de Redes [Fischetti et al., 2004; Rei et al., 2009; Martín & González, 2010].

4.3.1 Detalhes de Implementação

Apesar do Local Branching ser um método exato, capaz de encontrar a solução ótima global de um problema de Otimização Combinatória e prover um certificado para sua otimalidade, para Pochet & Wolsey [2006], o LB deve ser mais apropriadamente classificado como uma heurística de melhoria, já que seu objetivo é melhorar uma dada solução viável para um problema. Foi sob essa perspectiva que implementamos o LB para o PAGMGM, com o objetivo de tentar obter melhores limites superiores para as instâncias do problema não resolvidas pelo método BC.

A nossa implementação do algoritmo LB é baseada na formulação P_D^r. Considere que $(\overline{x}^r, \overline{y}) \in P_D^r \cap \mathbb{B}^{(2m+n)}$ denote uma solução inicial viável para o PAGMGM

e que $\overline{C}$ e $\overline{F} = V \setminus \overline{C}$ sejam, respectivamente, os conjuntos de vértices centrais e vértices folha associados à solução $(\overline{x}^r, \overline{y})$, isto é, $\overline{C} := \{i \in V : \overline{y}_i = 0\}$ e $\overline{F} := \{i \in V : \overline{y}_i = 1\}$. O esquema principal do método LB aplicado ao PAGMGM consiste em utilizar os conjuntos $\overline{C}$ e $\overline{F}$ para formular a restrição de Local Branching

$$\sum_{i \in \overline{C}} y_i + \sum_{i \in \overline{F}} (1 - y_i) \leq K, \tag{4.5}$$

em que o parâmetro K determina o tamanho da vizinhança da solução $(\overline{x}^r, \overline{y})$ que será investigada. Note que o tamanho da vizinhança limita a soma do número de vértices de $\overline{C}$ que podem trocar seu papel de central para folha e de $\overline{F}$ que podem trocar seu papel de folha para central em qualquer solução vizinha de $(\overline{x}^r, \overline{y})$. A restrição (4.5) é empregada para formular o primeiro subproblema do arcabouço de ramificação externo do Local Branching, dado por:

$$\min \left\{ \sum_{(i,j) \in A} c_{ij} x^r_{ij} : (x^r, y) \in P^r_D \cap (4.5) \cap (\mathbb{B}^{2m}, \mathbb{B}^n) \right\}. \tag{4.6}$$

Dizemos que uma solução $(\hat{x}^r, \hat{y}) \in P^r_D \cap \mathbb{B}^{(2m+n)}$ é aprimorante em relação à solução $(\overline{x}^r, \overline{y})$ se $\sum_{(i,j) \in A} c_{ij} \hat{x}^r_{ij} < \sum_{(i,j) \in A} c_{ij} \overline{x}^r_{ij}$. O subproblema (4.6), assim como todos os outros subproblemas formulados pelo método LB, são resolvidos com o algoritmo Branch-and-cut proposto na Seção 4.2. Em cada um desses subproblemas é imposto um limite de tempo *local* para a execução do BC. O algoritmo Local Branching inicia procurando resolver o subproblema (4.6). Quando esse subproblema é resolvido ou o limite de tempo local é atingido, quatro cenários distintos podem ocorrer:

(i) *O subproblema (4.6) é resolvido na otimalidade e sua solução ótima $(\hat{x}^r, \hat{y})$ é aprimorante em relação à solução $(\overline{x}^r, \overline{y})$.* Assuma que $\hat{C} := \{i \in V : \hat{y}_i = 0\}$ e $\hat{F} := \{i \in V : \hat{y}_i = 1\}$ e considere as restrições de Local Branching

$$\sum_{i \in \overline{C}} y_i + \sum_{i \in \overline{F}} (1 - y_i) \geq K + 1 \tag{4.7}$$

e

$$\sum_{i \in \hat{C}} y_i + \sum_{i \in \hat{F}} (1 - y_i) \leq K. \tag{4.8}$$

Um novo subproblema é formulado para investigar a vizinhança da solução aprimorante $(\hat{x}^r, \hat{y})$ ainda não explorada pelo subproblema (4.6), e em seguida

é resolvido. Esse novo subproblema é definido como:

$$\min \left\{ \sum_{(i,j)\in A} c_{ij} x_{ij}^r : (x^r, y) \in P_D^r \cap (4.7) \cap (4.8) \cap (\mathbb{B}^{2m}, \mathbb{B}^n) \right\}. \tag{4.9}$$

(ii) *O subproblema (4.6) é resolvido na otimalidade e nenhuma solução aprimorante é encontrada.* Neste caso é feita uma operação de diversificação, isto é, o tamanho da vizinhança da solução $(\overline{x}^r, \overline{y})$ a ser investigada é aumentado e um novo subproblema é então resolvido. Esse novo subproblema é dado por:

$$\min \left\{ \sum_{(i,j)\in A} c_{ij} x_{ij}^r : (x^r, y) \in P_D^r \cap (4.11) \cap (\mathbb{B}^{2m}, \mathbb{B}^n) \right\}, \tag{4.10}$$

em que a restrição de Local Branching

$$\sum_{i\in\overline{C}} y_i + \sum_{i\in\overline{F}} (1 - y_i) \leq K + \left\lceil \frac{K}{2} \right\rceil \tag{4.11}$$

é empregada. Limitamos o número máximo de operações de diversificação que podem ser feitas durante o algoritmo LB. Quando este limite é atingido, um último subproblema é formulado e resolvido. Este subproblema investiga todo o espaço de busca de soluções viáveis para o PAGMGM ainda não explorado pelos subproblemas anteriores.

(iii) *O limite de tempo local é atingido e uma solução aprimorante $(\hat{x}^r, \hat{y})$ é encontrada.* Neste caso, o mesmo procedimento descrito no cenário (i) é realizado, isto é, investigamos em seguida o subproblema (4.9).

(iv) *O limite de tempo local é atingido e nenhuma solução aprimorante é encontrada.* Com o objetivo de reduzir o tempo de busca do subproblema corrente, o tamanho da vizinhança da solução $(\overline{x}^r, \overline{y})$ é diminuído e um novo subproblema definido a partir da restrição de Local Branching

$$\sum_{i\in\overline{C}} y_i + \sum_{i\in\overline{F}} (1 - y_i) \leq \left\lceil \frac{K}{2} \right\rceil \tag{4.12}$$

é então investigado. Esse subproblema é dado por:

$$\min \left\{ \sum_{(i,j)\in A} c_{ij} x^r_{ij} : (x^r, y) \in P^r_D \cap (4.12) \cap (\mathbb{B}^{2m}, \mathbb{B}^n) \right\}. \tag{4.13}$$

Os quatro cenários apresentados podem ocorrer na investigação de qualquer um dos subproblemas formulados pelo arcabouço externo de ramificação do Local Branching, com exceção do último problema formulado, para o qual o BC é executado até que o certificado de otimalidade do PAGMGM seja obtido ou o limite de tempo *global* do LB seja atingido.

Em nosso estudo, empregamos o arcabouço externo de ramificação implementado em [Martinez & Cunha, 2009]. Nesse trabalho, foi proposto um arcabouço Local Branching genérico para a resolução de problemas de Programação Inteira, que permite que qualquer resolvedor de PI seja utilizado para resolver os subproblemas definidos pelo método LB. Essa característica é muito importante no nosso contexto, uma vez que necessitamos empregar o algoritmo Branch-and-cut como resolvedor interno. Além disso, o arcabouço permite definir as restrições de Local Branching a partir de qualquer conjunto de variáveis binárias, e controlar parâmetros como o tamanho de cada vizinhança, o número máximo de operações de diversificação e o limite de tempo para a resolução de cada subproblema. Todas essas características não estão disponíveis na implementação do método Local Branching oferecida pelo CPLEX.

Em nossos experimentos computacionais descritos ao final desta seção, adotamos o valor $K = 5$ para todas as instâncias testadas. Esse valor foi escolhido após testes preliminares indicarem que mesmo para as instâncias de tamanho médio consideradas, valores maiores para K resultavam em subproblemas quase nunca resolvidos na otimalidade pelo BC, no limite de tempo imposto. Utilizamos um número máximo de 3 operações de diversificação. Para cada instância avaliada, estabelecemos um limite de tempo global de 21600 segundos para a execução do LB e um limite de tempo local de 7200 segundos para a resolução de cada subproblema por meio do BC.

Para as instâncias da classe *ALM*, de maneira similar à abordagem utilizada para o BC, empregamos a melhor solução encontrada pela heurística CMP, após 300 segundos de processamento, para formular a restrição de Local Branching (4.5) e o primeiro subproblema (4.6). Por outro lado, para as instâncias das classes *CRD* e *SYM*, formulamos a restrição (4.5) a partir da primeira solução viável encontrada

pelo método BC. Dessa forma, para todas as 90 instâncias avaliadas, inicializamos o LB a partir da primeira solução viável considerada pelo BC.

4.3.2 Resultados Computacionais

Nesta seção, apresentamos os resultados computacionais obtidos pelo método LB, e os comparamos aos resultados fornecidos pelos métodos BB-MTZ e BC. Optamos por não comparar o LB às meta-heurísticas HVNS, uma vez que, para todas as instâncias consideradas, os melhores limites superiores apresentados em [Martins & Souza, 2009] foram mais fracos que aqueles obtidos pelo método BB-MTZ ou pelo algoritmo BC.

A Tabela 4.6 sumariza os resultados obtidos pelos três métodos para as 90 instâncias de teste avaliadas. As duas primeiras colunas indicam, respectivamente, o valor de n e o número de instâncias testadas com n vértices. Os resultados obtidos pelo BB-MTZ são apresentadas nas três colunas subsequentes, a saber: o número de instâncias resolvidas pelo método (resol.), o tempo total gasto para encontrar a solução ótima para as instâncias resolvidas ($tt_{\overline{w}}$, em segundos) e o número de melhores limites superiores obtidos iguais aos melhores limites superiores conhecidos ($\overline{w} = \overline{w}^*$). Assim sendo, essa última coluna indica em quantos casos o algoritmo BB-MTZ forneceu um limite superior idêntico ao melhor conhecido, considerando os resultados obtidos pelos métodos BB-MTZ, BC e LB. As colunas seguintes apresentam os mesmos dados (resol., $tt_{\overline{w}}$ e $\overline{w} = \overline{w}^*$) para os métodos BC e LB. De forma análoga à convenção adotada anteriormente, para os valores de n em que os métodos não foram capazes de resolver nenhuma instância, o símbolo "-" é apresentado nas entradas correspondentes das colunas $tt_{\overline{w}}$.

Tabela 4.6. Síntese dos resultados obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], BC e LB, para as instâncias das classes *CRD*, *SYM* e *ALM*. Todos os métodos foram testados no mesmo ambiente computacional.

		BB-MTZ			BC			LB		
n	#instâncias	resol.	$tt_{\overline{w}}$	$\overline{w} = \overline{w}^*$	resol.	$tt_{\overline{w}}$	$\overline{w} = \overline{w}^*$	resol.	$tt_{\overline{w}}$	$\overline{w} = \overline{w}^*$
30	12	12	300,6	12	12	4,4	12	12	8,1	12
50	18	18	21364,4	18	18	499,4	18	18	411,5	18
70	18	12	1293,9	15	18	278,5	18	18	446,0	18
100	12	6	42969,3	6	6	4356,5	6	6	2029,3	12
200	6	0	-	0	0	-	3	0	-	3
300	6	0	-	0	0	-	1	0	-	5
400	9	0	-	2	0	-	2	0	-	5
500	9	0	-	2	0	-	3	0	-	4

A Tabela 4.6 mostra que o método LB foi capaz de resolver o mesmo número

de instâncias que o BC e 6 a mais que o BB-MTZ. O tempo total gasto pelo LB para encontrar as melhores soluções viáveis para as instâncias resolvidas pelos três métodos corresponde a $4{,}39\%$ do tempo gasto pelo BB-MTZ e $56{,}33\%$ do tempo gasto pelo BC. Como pode ser observado na tabela, essa diferença nos tempos computacionais gastos pelos métodos LB e BC foi determinada pelas maiores instâncias resolvidas, definidas em grafos com 100 vértices. Para as instâncias menores, não houve dominância de um método sobre o outro em termos de tempos computacionais.

Considerando as 36 instâncias para as quais a otimalidade não foi provada, o método LB forneceu 23 novos melhores limites superiores. Para as outras 13 instâncias em aberto, 9 melhores limites superiores foram obtidos com o método BC e 4 com o método BB-MTZ. Em particular, para todas as 6 instâncias de 100 vértices não resolvidas por nenhum dos três métodos, os melhores limites superiores foram obtidos pelo algoritmo LB.

4.3.2.1 Resultados Detalhados

Nesta seção, apresentamos os resultados detalhados dos experimentos realizados com os métodos BB-MTZ, BC e LB.

As Tabelas 4.7 e 4.8 apresentam os resultados computacionais obtidos pelos três métodos. A primeira coluna delas indica os ids das instâncias consideradas. Cada um dos três blocos de três colunas seguintes mostram os resultados obtidos, respectivamente, pelos métodos BB-MTZ, BC e LB. São apresentados para cada método: o melhor limite superior ($\overline{w}$) obtido no limite de tempo de 6 horas, o tempo computacional necessário para obtê-lo ($t_{\overline{w}}$, em segundos) e o tempo total de execução do algoritmo (t, em segundos).

Os limites inferiores obtidos pelo método LB foram mais fracos que aqueles fornecidos pelo BC para todas as instâncias não resolvidas, e por isso foram omitidos nas tabelas. Essa baixa qualidade dos limites duais dados pelo LB pode ser explicada pelo fato de termos implementado o método sob a perspectiva de uma heurística de melhoria, com o objetivo de obter limites primais mais fortes para o PAGMGM. Assim, devido ao desbalanceamento da árvore externa do LB e ao limite de tempo global de 6 horas imposto para a execução do método, o tempo disponível para a resolução do último subproblema formulado pelo LB, em geral, não foi suficiente para que limites duais mais fortes fossem obtidos.

Assim como o algoritmo BC, o LB claramente domina o BB-MTZ em termos de tempos computacionais. Observe que, para algumas instâncias, o tempo gasto pelo LB provar a otimalidade foi duas ou três ordens de grandeza menor que o tempo

correspondente gasto pelo BB-MTZ. Em apenas 5 casos (instâncias com ids 29, 33, 38, 46 e 47), o BB-MTZ foi capaz de provar a otimalidade mais rapidamente que o LB. No total, considerando todas as 48 instâncias resolvidas pelos dois métodos, o LB gastou $17{,}97\%$ do tempo gasto pelo BB-MTZ para resolvê-las.

Como pode ser notado nas Tabelas 4.7 e 4.8, o LB resolveu na otimalidade as mesmas instâncias que o método BC, tendo encontrado as soluções ótimas para essas instâncias em $34{,}73\%$ do tempo total gasto pelo BC para encontrá-las. Em contrapartida, o LB gastou, em geral, maiores tempos computacionais para provar a otimalidade dessas soluções. Considerando todas as instâncias resolvidas pelos dois métodos, o tempo total gasto pelo LB para prover os certificados de otimalidade foi $40{,}46\%$ superior ao tempo correspondente gasto pelo BC. Em apenas três casos (instâncias com ids 16, 20 e 52), o tempo gasto pelo LB para resolver a instância foi inferior ao tempo gasto pelo BC.

Cabe ressaltar que, assim como ocorreu com o método BC, o LB não foi capaz de encontrar uma solução viável melhor que a solução inicial fornecida pela heurística CMP para três instâncias, todas de 500 vértices (ids 82, 83 e 87). Em todos esses casos, a solução fornecida pela heurística CMP era pior que as melhores soluções encontradas pelos métodos BB-MTZ e BC após 6 horas de processamento.

Tabela 4.7. Comparação entre os algoritmos BB-MTZ [Akgún & Tansel, 2010], BC e LB, para as instâncias das classes *CRD* e *SYM*.

	BB-MTZ			BC			LB		
id	$\overline{w}$	$t_{\overline{w}}$	t	$\overline{w}$	$t_{\overline{w}}$	t	$\overline{w}$	$t_{\overline{w}}$	t
1	4026	3,0	5,5	4026	0,2	0,2	4026	0,2	0,5
2	3793	278,8	329,4	3793	0,1	0,1	3793	0,7	1,0
3	4293	1,9	2,1	4293	0,2	0,2	4293	0,3	0,6
4	5026	2,2	3,3	5026	0,7	0,7	5026	1,1	1,6
5	4648	5,8	5,8	4648	0,2	0,2	4648	0,0	0,5
6	5425	5,8	7,5	5425	1,7	2,5	5425	4,4	6,7
7	5512	18153,2	20283,9	5512	3,3	3,9	5512	4,4	8,5
8	5813	828,7	6920,1	5813	49,8	50,2	5813	15,1	52,0
9	5590	398,5	602,7	5590	2,4	3,0	5590	4,7	6,8
10	6908	457,3	3150,4	6908	70,0	70,6	6908	83,6	85,7
11	7204	786,0	944,4	7204	68,3	77,7	7204	36,6	81,0
12	7277	651,0	3713,2	7277	269,1	274,3	7277	226,4	288,0
13	9633	21,3	23,3	9633	8,3	8,8	9633	11,4	19,2
14	9743	10,6	16,7	9743	3,8	5,7	9743	5,7	9,7
15	9855	0,7	18,1	9855	4,5	5,1	9855	4,4	10,0
16	6516	1096,8	21600,0	6516	7,2	118,8	6516	20,5	110,3
17	6621	1731,3	21600,0	6586	2288,3	2299,5	6586	3164,9	3177,4
18	7066	20784,9	21600,0	7053	2062,0	2065,6	7053	1626,3	3130,8
19	8134	19688,4	21600,0	8134	15309,0	15316,9	8134	149,0	16738,3
20	7943	355,6	21600,0	7943	1139,9	1143,6	7943	44,3	1118,7
21	8421	13324,4	21600,0	8419	1049,1	1198,4	8419	1475,9	1500,0
22	11235	141,5	143,4	11235	62,1	66,1	11235	136,0	137,0
23	11373	295,0	632,9	11373	47,0	180,2	11373	127,5	361,4
24	11979	52,5	69,7	11979	16,1	18,2	11979	24,4	36,8
25	9334	19875,7	21600,0	9349	20177,9	21600,0	9251	1988,2	21600,0
26	9562	21510,5	21600,0	9571	9333,1	21600,0	9557	563,2	21600,0
27	9635	21272,7	21600,0	9653	16813,7	21600,0	9622	1346,6	21600,0
28	12916	10089,3	18939,7	12916	1051,1	1622,7	12916	622,5	4021,6
29	13026	565,4	833,9	13026	443,7	445,0	13026	577,9	856,6
30	13365	6867,0	7974,3	13365	322,0	577,5	13365	91,7	1962,7
31	1197	0,2	0,2	1197	0,1	0,1	1197	0,0	0,2
32	1435	0,1	0,2	1435	0,1	0,1	1435	0,1	0,1
33	1408	0,0	0,1	1408	0,0	0,0	1408	0,1	0,1
34	1765	1,6	1,6	1765	0,7	0,7	1765	0,6	0,7
35	2090	1,1	1,1	2090	0,2	0,2	2090	0,3	0,5
36	2008	0,1	0,6	2008	0,1	0,1	2008	0,2	0,2
37	1278	2,1	2,1	1278	0,5	0,5	1278	0,9	0,9
38	1178	0,3	0,9	1178	0,5	0,5	1178	0,6	0,9
39	1615	1,0	1,0	1615	0,2	0,2	1615	0,3	0,5
40	2054	8,3	10,5	2054	5,1	5,2	2054	5,6	6,0
41	1760	2,8	3,6	1760	0,9	0,9	1760	1,1	1,7
42	2525	9,5	20,2	2525	6,4	7,0	2525	5,3	9,5
43	4121	7,7	7,8	4121	2,5	2,5	4121	1,6	2,9
44	4166	13,0	14,1	4166	1,0	1,5	4166	1,4	2,5
45	4979	12,5	24,1	4979	3,0	3,8	4979	2,4	6,5
46	1360	3,6	3,6	1360	1,3	1,3	1360	2,1	4,2
47	1448	9,0	9,2	1448	3,9	4,1	1448	9,8	9,9
48	1521	23,7	24,2	1521	4,3	4,4	1521	3,7	7,7
49	2028	25,1	25,2	2028	6,9	8,9	2028	10,2	15,0
50	2165	31,1	34,0	2165	1,9	1,9	2165	2,1	3,4
51	2210	38,8	47,4	2210	19,4	19,4	2210	16,4	21,5
52	4979	162,6	163,9	4979	62,8	63,5	4979	7,1	28,7
53	4787	194,6	238,0	4787	29,4	36,1	4787	36,9	53,9
54	4997	316,5	421,0	4997	23,5	50,4	4997	69,8	100,5

Tabela 4.8. Comparação entre os algoritmos BB-MTZ [Akgún & Tansel, 2010], BC e LB, para as instâncias da classe *ALM*.

	BB-MTZ			BC			LB		
id	$\overline{w}$	$t_{\overline{w}}$	t	$\overline{w}$	$t_{\overline{w}}$	t	$\overline{w}$	$t_{\overline{w}}$	t
55	5457	8185,7	21600,0	5353	6408,0	21600,0	5317	1145,4	21600,0
56	5197	20448,5	21600,0	5055	7731,7	21600,0	5022	1147,8	21600,0
57	5450	21317,5	21600,0	5450	20656,4	21600,0	5428	1106,1	21600,0
58	7164	2113,5	2269,8	7164	130,2	158,3	7164	156,5	445,8
59	6886	5204,2	5712,4	6886	853,3	857,8	6886	252,4	1950,3
60	7382	18129,9	20247,4	7382	1556,2	3581,3	7382	328,3	6254,8
61	7217	21484,8	21600,0	7075	16366,6	21600,0	7122	11190,1	21600,0
62	7180	20379,7	21600,0	7245	19397,2	21600,0	7118	10558,0	21600,0
63	7556	21368,5	21600,0	7492	1222,2	21600,0	7514	15607,2	21600,0
64	9680	21458,7	21600,0	9652	12749,4	21600,0	9764	19436,9	21600,0
65	9916	21561,0	21600,0	9820	9850,8	21600,0	9801	19569,0	21600,0
66	10039	21576,6	21600,0	9978	8539,7	21600,0	9904	19994,1	21600,0
67	9211	21595,6	21600,0	9705	1748,8	21600,0	9195	18335,0	21600,0
68	9295	21186,5	21600,0	9407	1522,8	21600,0	8877	21371,8	21600,0
69	9359	21354,7	21600,0	9458	3133,5	21600,0	9296	21419,4	21600,0
70	12603	20440,4	21600,0	12891	3417,1	21600,0	12381	20256,7	21600,0
71	12723	21039,6	21600,0	12246	1935,5	21600,0	12272	17798,8	21600,0
72	13063	21599,8	21600,0	13110	2144,1	21600,0	12725	17976,4	21600,0
73	10917	21326,1	21600,0	11079	5241,3	21600,0	10670	21354,3	21600,0
74	10791	20900,3	21600,0	11036	4670,7	21600,0	11076	13916,8	21600,0
75	10336	21487,9	21600,0	10711	10746,7	21600,0	10447	20489,2	21600,0
76	15055	21587,5	21600,0	14611	12527,7	21600,0	14705	19950,7	21600,0
77	14774	21086,9	21600,0	15086	5585,1	21600,0	14679	19477,8	21600,0
78	15142	21376,5	21600,0	14506	5895,1	21600,0	14184	18370,9	21600,0
79	23287	19095,7	21600,0	21260	14805,9	21600,0	20345	20089,1	21600,0
80	23951	3072,8	21600,0	20473	12807,5	21600,0	20520	20121,8	21600,0
81	22445	3268,1	21600,0	21478	300,0	21600,0	20427	12783,4	21600,0
82	11872	19183,2	21600,0	11881	15773,7	21600,0	13328	300,0	21600,0
83	12512	20818,5	21600,0	12518	14871,7	21600,0	12990	300,0	21600,0
84	12190	21336,5	21600,0	12035	12348,7	21600,0	12562	10692,2	21600,0
85	17900	20405,3	21600,0	15617	20011,0	21600,0	16187	21201,7	21600,0
86	33992	1190,5	21600,0	16344	13718,6	21600,0	16325	20336,5	21600,0
87	17168	21198,6	21600,0	16466	15514,9	21600,0	18644	300,0	21600,0
88	42194	1796,0	21600,0	24508	300,0	21600,0	23465	21377,1	21600,0
89	23871	5105,2	21600,0	24900	300,0	21600,0	23668	20057,4	21600,0
90	53380	1873,0	21600,0	23901	20303,5	21600,0	22726	19029,6	21600,0

4.4 Comentários

Os algoritmos Branch-and-cut e Local Branching, propostos neste capítulo para a resolução exata do PAGMGM, permitiram aprimorar em alguns aspectos os melhores resultados conhecidos na literatura do problema. Os métodos forneceram novos certificados de otimalidade para 6 instâncias do PAGMGM, e foram capazes de resolver 54 das 90 instâncias de teste consideradas neste estudo. Além disso, os tempos computacionais gastos para a resolução dessas instâncias foram entre uma e duas ordens de grandeza menores que os tempos gastos pelo algoritmo proposto em [Akgún & Tansel, 2010]. Considerando as 36 instâncias para as quais a otimalidade não foi provada, o algoritmo BC forneceu 34 novos melhores limites inferiores e 9 novos melhores limites superiores. Já o método LB foi capaz de obter 23 novos melhores limites superiores.

Apesar desses resultados obtidos pelos métodos BC e LB, percebemos que o PAGMGM é ainda muito difícil de ser resolvido, principalmente para as maiores instâncias consideradas nesta dissertação. Em particular, os métodos BC e LB não foram capazes de resolver nenhuma instância com 200 ou mais vértices. Para esses casos, os *gaps* de dualidade obtidos pelo BC foram elevados, com um valor médio superior a 13%.

Dada a dificuldade em se resolver as maiores instâncias do PAGMGM por meio dos algoritmos apresentados neste capítulo, baseados na formulação P_D^r, propomos uma Relaxação Lagrangeana baseada na formulação P_I. Essa formulação fornece os limites de Programação Linear mais fortes que conhecemos para o problema. Esperamos com essa abordagem conseguir avaliar mais rapidamente $w(P_I)$, por meio de um algoritmo paralelo para resolver o Problema Dual Lagrangeano. Pelo lado primal, desenvolvemos uma Heurística Lagrangeana com o objetivo de obter melhores limites superiores para o PAGMGM. No capítulo seguinte discutimos esses métodos.

Capítulo 5

Algoritmos Sequenciais e Paralelos para o PAGMGM Baseados em Relaxação Lagrangeana

Neste capítulo, introduzimos algoritmos sequenciais e paralelos para o PAGMGM baseados em Relaxação Lagrangeana. O nosso principal objetivo é conseguir avaliar, ou ao menos aproximar, o limite inferior $w(P_I)$ de forma mais rápida que por meio do algoritmo de planos de corte discutido na Seção 3.5.1 e pela nossa implementação do procedimento iterativo proposto em [Gouveia & Telhada, 2011]. Para tanto, apresentamos primeiramente um esquema de Relaxação Lagrangeana obtido a partir da reformulação por interseção. Em seguida, descrevemos uma implementação paralela de uma variação do Método de Subgradiente empregada para resolver o Problema Dual Lagrangeano. Apresentamos então uma Heurística Lagrangeana para encontrar soluções viáveis de boa qualidade para o PAGMGM. Por fim, avaliamos computacionalmente os métodos propostos neste capítulo.

5.1 Relaxação Lagrangeana da Reformulação por Interseção

Muitos problemas difíceis de Programação Inteira podem ser vistos como problemas fáceis de otimização aos quais restrições complicantes são adicionadas. Assim, ao se relaxar tais restrições, obtemos um problema de fácil resolução, cuja solução ótima fornece um limite inferior (no caso de problemas de minimização) válido para o problema original. A Relaxação Lagrangeana (RL) é uma técnica de

decomposição que explora essa ideia, e pode ser empregada como uma alternativa à relaxação de Programação Linear, para avaliar (ou aproximar) os limites inferiores de um problema de PI, sem resolver explicitamente Programas Lineares.

Nesta seção, introduzimos uma Relaxação Lagrangeana baseada na reformulação por interseção para o PAGMGM, cujo objetivo é obter uma boa aproximação de $w(P_I)$ em tempos computacionais razoáveis. Considerando o que foi exposto na Seção 3.3, reescrevemos a seguir a reformulação por interseção como apresentada na Seção 3.5.1.3. Assuma que o poliedro $P_I \subset \mathbb{R}^{m(2n+1)+n}$ seja definido por:

$$\sum_{i \in V}(1 - y_i) \leq \left\lfloor \frac{n-2}{d-1} \right\rfloor, \tag{5.1}$$

$$x^r(A) = n - 1, \ \ \forall r \in V, \tag{5.2}$$

$$x^r(\delta^-(i)) = 1, \ \ \forall r \in V, \ \ i \in V \setminus \{r\}, \tag{5.3}$$

$$(1 - d)y_i - z(\delta(i)) \leq -d, \ \ \forall i \in V, \tag{5.4}$$

$$(n - 2)y_i + z(\delta(i)) \leq n - 1, \ \ \forall i \in V, \tag{5.5}$$

$$z_{ij} - x^r_{ij} - x^r_{ji} = 0, \ \ \forall r \in V, \ \ \{i, j\} \in E, \tag{5.6}$$

$$z_{ij} + y_i + y_j \leq 2, \ \ \forall \{i, j\} \in E, \tag{5.7}$$

$$x^r_{ij} + y_i \leq 1, \ \ \forall r \in V, \ \ (i, j) \in A \setminus \delta^+(r), \tag{5.8}$$

$$z_{ij} \geq 0, \ \ \forall \{i, j\} \in E, \tag{5.9}$$

$$0 \leq y_i \leq 1, \ \ \forall i \in V, \tag{5.10}$$

$$0 \leq x^r_{ij} \leq 1, \ \ \forall r \in V, \ \ (i, j) \in A. \tag{5.11}$$

Vamos relaxar e dualizar as restrições (5.4)-(5.8). Para tanto, assuma que os multiplicadores $\{\alpha_i \in \mathbb{R}_+ : i \in V\}$, $\{\beta_i \in \mathbb{R}_+ : i \in V\}$, $\{\gamma^r_{ij} \in \mathbb{R} : r \in V, \ \{i, j\} \in E\}$, $\{\omega_{ij} \in \mathbb{R}_+ : \{i, j\} \in E\}$ e $\{\mu^r_{ij} \in \mathbb{R}_+ : r \in V, \ (i, j) \in A \setminus \delta^+(r)\}$ são associados, respectivamente, às restrições (5.4), (5.5), (5.6), (5.7) e (5.8). Por simplicidade, assuma que $\pi = (\alpha, \beta, \gamma, \omega, \mu)$ denota o vetor de multiplicadores Lagrangeanos de dimensão apropriada. Assuma também que $\{\overline{c}_{ij} : \{i, j\} \in E\}$, $\{\overline{u}_i : i \in V\}$ e $\{\overline{l}^r_{ij} : r \in V, \ (i, j) \in A\}$ denotem os custos Lagrageanos associados às variáveis z, y e x, respectivamente.

Esse esquema de relaxação resulta em um Problema de Relaxação Lagrangeana (PRL) que pode ser decomposto em $n + 2$ subproblemas independentes: um envolvendo a variável z, outro envolvendo a variável y e um envolvendo cada variável $\{x^r : r \in V\}$. Dessa forma, limites inferiores para o PAGMGM podem ser

obtidos ao se resolver o PRL dado por

$$w(\pi) = w_z(\pi) + w_y(\pi) + \sum_{r \in V} w_r(\pi) + const(\pi), \tag{5.12}$$

em que $const(\pi)$ é um termo constante independente de z, y e x, definido por

$$const(\pi) = d\sum_{i \in V} \alpha_i + (1-n)\sum_{i \in V} \beta_i - 2\sum_{\{i,j\} \in E} \omega_{ij} - \sum_{r \in V} \sum_{(i,j) \in A \setminus \delta^+(r)} \mu_{ij}^r, \tag{5.13}$$

e w_z, w_y e w_r são, respectivamente, os valores ótimos dos problemas (5.14), (5.15) e (5.16):

$$w_z(\pi) = \min \left\{ \sum_{\{i,j\} \in E} \overline{c}_{ij} z_{ij} : z \in T \right\}, \tag{5.14}$$

$$w_y(\pi) = \min \left\{ \sum_{i \in V} \overline{u}_i y_i : y \in \mathbb{B}^n, \ \ y \text{ satisfaz (5.1)} \right\}, \tag{5.15}$$

$$w_r(\pi) = \min \left\{ \sum_{(i,j) \in A} \overline{l}_{ij}^r x_{ij}^r : x^r \in \mathbb{B}^{2m}, \ \ x^r \text{ satisfaz (5.2)-(5.3)} \right\}. \tag{5.16}$$

Em (5.14), T denota qualquer descrição genérica da envoltória convexa do Politopo das Árvores Geradoras de G, por exemplo, $T = \{z \in \mathbb{R}_+^m : z(E) = n-1, z(E(S)) \leq |S|-1, \forall S \subset V, S \neq \emptyset\}$. Apesar das restrições $z \in T$ serem redundantes para o PRL (5.12), optamos por utilizá-las em (5.14) com o objetivo de reforçar a topologia de rede nas soluções ótimas do PRL.

Para resolver (5.14), precisamos apenas encontrar uma árvore geradora de custo mínimo do grafo G, valorado com custos Lagrangeanos $\{\overline{c}_{ij} : \{i,j\} \in E\}$ associados às suas arestas. Tal árvore pode ser encontrada em $O(m \log n)$, por exemplo, com o algoritmo proposto em [Kruskal, 1956]. Para resolver (5.15), implementamos um procedimento com complexidade de tempo $O(n \log n)$, que ordena os vértices de V de acordo com os custos Lagrangeanos $\{\overline{u}_i : i \in V\}$. Assim, atribuímos $y_i = 1$ às entradas de y associadas aos $n - \left\lfloor \frac{n-2}{d-1} \right\rfloor$ menores valores $\overline{u}_i$. Para as demais entradas de y, atribuímos $y_i = 1$ se $\overline{u}_i < 0$ e $y_i = 0$ caso contrário. Cada subproblema (5.16) pode ser resolvido em $O(m)$ por inspeção, selecionando o arco $(i,j) \in \delta^-(j)$ associado ao menor valor $\overline{l}_{ij}^r$, para todo $j \in V$.

Uma vez que $w(\pi)$ nos fornece um limite inferior válido para o $w(P_I)$, é desejável encontrar os valores dos multiplicadores Lagrangeanos π que maximizam esse

limite. Para tanto, resolvemos o Problema Dual Lagrangeano (DL):

$$w_{DL} = \max_{\pi} w(\pi). \tag{5.17}$$

Observe que os subproblemas (5.14)-(5.16) satisfazem a Propriedade da Integralidade [Geoffrion, 1974], e portanto, $w_{DL} = w(P_I)$. Para resolver o problema (5.17), ou ao menos obter uma boa aproximação de w_{DL}, implementamos uma variação do Método de Subgradiente, discutida na próxima seção.

5.2 Otimização Via Método de Subgradiente

Vários métodos podem ser empregados para resolver o DL. Pelos bons resultados obtidos em muitos problemas práticos e por sua simplicidade de implementação, o Método de Subgradiente (MS) [Held & Karp, 1970, 1971; Held et al., 1974] é uma das técnicas mais empregadas. Outras técnicas comumente utilizadas são o Método de Feixes [Lemarechal, 1974] e o Método do Volume [Barahona & Anbil, 2000; Bahiense et al., 2002, 2003].

Nesta dissertação, implementamos uma variação do MS proposta em [Camerini et al., 1975], denominada *Deflected Subgradient Method* (DSM). O DSM é um método iterativo cuja única diferença em relação ao MS clássico está relacionada à escolha da direção de busca. Assuma que em uma dada iteração k do DSM, todos os valores (multiplicadores, custos Lagrangeanos, subgradientes, etc) envolvidos no método são indexados por k. A seguir, sintetizamos os principais passos do DSM aplicado para a resolução de (5.17).

Inicialização

1. Inicialize o contador de iterações: $k \leftarrow 0$.
2. Atribua valores iniciais válidos aos multiplicadores Lagrangeanos, por exemplo:
   a) $\alpha_i^k = 0, \ \forall i \in V$,
   b) $\beta_i^k = 0, \ \forall i \in V$,
   c) $\gamma_{ij}^{r,k} = 0, \ \forall r \in V, \ \{i,j\} \in E$,
   d) $\omega_{ij}^k = 0, \ \forall \{i,j\} \in E$,
   e) $\mu_{ij}^{r,k} = 0, \ \forall r \in V, \ (i,j) \in A \setminus \delta^+(r)$.

Repita

1. Calcule os *custos Lagrangeanos*:
   a) $\overline{c}_{ij}^k = c_{ij} - (\alpha_i^k + \alpha_j^k) + (\beta_i^k + \beta_j^k) + \sum_{r \in V} \gamma_{ij}^{r,k} + \omega_{ij}^k, \ \ \forall \{i,j\} \in E,$
   b) $\overline{u}_i^k = (1-d)\alpha_i^k + (n-2)\beta_i^k + \sum_{\{i,j\} \in \delta(i)} \omega_{ij}^k + \sum_{r \in V \setminus \{i\}} \sum_{(i,j) \in \delta^+(i)} \mu_{ij}^{r,k}, \ \ \forall i \in V,$
   c) $\overline{l}_{ij}^{r,k} = \begin{cases} -\gamma_{ij}^{r,k} & \text{se} \quad i = r, \\ -\gamma_{ij}^{r,k} + \mu_{ij}^{r,k} & \text{se} \quad i \neq r, \end{cases} \qquad \forall r \in V, \ \ (i,j) \in A.$
2. Resolva os *subproblemas* (5.14)-(5.16). Sejam $\overline{z}^k$, $\overline{y}^k$ e $\{\overline{x}^{r,k} : r \in V\}$ as suas soluções ótimas, respectivamente.
3. Calcule um *vetor subgradiente* $s^k = (s_{\alpha_i^k}, s_{\beta_i^k}, s_{\gamma_{ij}^{rk}}, s_{\omega_{ij}^k}, s_{\mu_{ij}^{r,k}})$ da função DL (5.17), avaliada em $\overline{z}^k$, $\overline{y}^k$ e $\{\overline{x}^{r,k} : r \in V\}$:
   a) $s_{\alpha_i^k} = (1-d)\overline{y}_i^k - \overline{z}^k(\delta(i)) + d, \ \ \forall i \in V,$
   b) $s_{\beta_i^k} = (n-2)\overline{y}_i^k + \overline{z}^k(\delta(i)) - n + 1, \ \ \forall i \in V,$
   c) $s_{\gamma_{ij}^{r,k}} = \overline{z}_{ij}^k - \overline{x}_{ij}^{r,k} - \overline{x}_{ji}^{r,k}, \ \ \forall r \in V, \ \ \{i,j\} \in E,$
   d) $s_{\omega_{ij}^k} = \overline{z}_{ij}^k + \overline{y}_i^k + \overline{y}_j^k - 2, \ \ \forall \{i,j\} \in E,$
   e) $s_{\mu_{ij}^{r,k}} = \overline{x}_{ij}^{r,k} + \overline{y}_i^k - 1, \ \ \forall r \in V, \ \ (i,j) \in A \setminus \delta^+(r).$
4. Calcule um *vetor de direção de busca* b^k:

$$b^k = \begin{cases} s^k & \text{se} \quad k = 0, \\ (1-\lambda)s^k + \lambda b^{k-1} & \text{se} \quad k \geq 1, \end{cases}$$

   em que $\lambda \in (0,1)$ é um escalar real.
5. Calcule o *tamanho do passo* ρ^k:

$$\rho^k = \frac{\varepsilon(UB - w(\alpha^k, \beta^k, \gamma^k, \omega^k, \mu^k))}{||b^k||^2},$$

   em que $\varepsilon \in (0,2]$ é um escalar real, UB é um limite superior qualquer para w e $||b^k||$ é a norma Euclidiana do vetor de direção de busca b^k.
6. Atualize os *multiplicadores Lagrangeanos*:
   a) $\alpha_i^{k+1} = \max\{\alpha_i^k + \rho^k b_{\alpha_i^k}, 0\}, \ \ \forall i \in V,$
   b) $\beta_i^{k+1} = \max\{\beta_i^k + \rho^k b_{\beta_i^k}, 0\}, \ \ \forall i \in V,$
   c) $\gamma_{ij}^{r,k+1} = \gamma_{ij}^{r,k} + \rho^k b_{\gamma_{ij}^{r,k}}, \ \ \forall r \in V, \ \ \{i,j\} \in E,$
   d) $\omega_{ij}^{k+1} = \max\{\omega_{ij}^k + \rho^k b_{\omega_{ij}^k}, 0\}, \ \ \forall \{i,j\} \in E,$
   e) $\mu_{ij}^{r,k+1} = \max\{\mu_{ij}^{r,k} + \rho^k b_{\mu_{ij}^{r,k}}, 0\}, \ \ \forall r \in V, \ \ (i,j) \in A \setminus \delta^+(r).$
7. Avalie a *convergência* do método de acordo com as regras:

a) Se $\varepsilon < \texttt{min}_\varepsilon$ ou

b) $k = \texttt{max}_k$ ou

c) o limite de tempo `lt(DSM)` foi atingido, pare.

Em todas as iterações do Método de Subgradiente clássico, o vetor de direção de busca é um vetor subgradiente da função Dual Lagrangeana, isto é, $b^k = s^k$ para $k = 0, \ldots, \texttt{max}_k - 1$. Observe que no método proposto em [Camerini et al., 1975], apenas na primeira iteração temos $b^0 = s^0$. Para todas as demais iterações ($k > 0$), o cálculo da direção de busca considera todas as direções de busca empregadas nas iterações $0, \ldots, k-1$, fazendo $b^k = (1-\lambda)s^k + \lambda b^{k-1}$, em que λ é um escalar real no intervalo $(0, 1)$. Após experimentos preliminares, decidimos utilizar em nossa implementação o valor $\lambda = 0{,}05$.

Com relação aos demais parâmetros do DSM, inicializamos $\varepsilon = 2{,}0$ e atualizamos $\varepsilon = 0{,}8\varepsilon$ sempre que 200 iterações do método são realizadas sem aprimorar o melhor limite Dual Lagrangeano até então obtido. Quando $\varepsilon < \texttt{min}_\varepsilon = 10^{-4}$, o DSM é finalizado. Os outros parâmetros utilizados para a avaliação da convergência do método foram $\texttt{max}_k = 20000$ e `lt(DSM)` $= 14400$ segundos (4 horas).

É sabido que métodos de subgradientes tendem a apresentar dificuldades de convergência se muitas restrições de igualdade são dualizadas [Beasley, 1993]. Isso acontece já que, nesse caso, mais entradas do vetor de direção de busca tendem a assumir valores não nulos, o que aumenta a norma Euclidiana $||b^k||$ e reduz para valores pequenos o tamanho do passo ρ^k. Isso posto, para o caso particular do PAGMGM, encontramos dificuldades adicionais na resolução do Problema Dual Lagrangeano através do DSM, já que o número de restrições de igualdade dualizadas é da ordem de $O(n^3)$ para instâncias definidas em grafos completos, como as utilizadas nesta dissertação. Uma consequência do número elevado de restrições de igualdade dualizadas é o aumento do número de iterações necessárias para que o DSM consiga prover uma boa aproximação de w_{LD}. Além disso, como o número de variáveis da reformulação por interseção é também da ordem de $O(n^3)$, o tempo de computação de cada iteração do DSM é bastante alto.

Por outro lado, o DSM possui fontes de paralelismo que podem ser exploradas com o objetivo de reduzir o tempo computacional gasto em cada iteração. Apesar de métodos de subgradientes serem essencialmente sequenciais, cada operação, como por exemplo, o cálculo dos custos Lagrangeanos e dos subgradientes, pode ser feita através de laços paralelos. Outras operações como a atualização dos multiplicadores Lagrangeanos e o cálculo das direções de busca envolvem a multiplicação de vetores por um escalar e a soma de vetores. Além disso, a resolução dos sub-

problemas Lagrangeanos (5.14)-(5.16) pode ser feita de forma independente. Todas essas operações são passíveis de paralelização, como descrevemos a seguir.

5.3 Uma Implementação Paralela do Método de Subgradiente

Com o objetivo de executar mais iterações do DSM no limite de tempo computacional `lt(DSM)` e obter melhores aproximações de $w(P_I)$, implementamos uma versão paralela do DSM. Antes de descrevermos essa implementação, revisamos a seguir os principais conceitos básicos de programação paralela empregados no texto. Para uma descrição mais completa desses e de outros conceitos relacionados à programação paralela, sugerimos o livro texto [Rauber & Rünger, 2010].

5.3.1 Conceitos Básicos de Programação Paralela

Há alguns anos, fabricantes de circuitos integrados passaram a produzir processadores com várias unidades de computação independentes em um único chip. Usualmente, o termo *núcleo de processamento* (*core*) é utilizado para referenciar uma única unidade de computação. Assim, um processador *single-core* possui apenas um núcleo, enquanto o termo *multi-core* é utilizado para denotar um processador com vários núcleos. Atualmente, computadores pessoais comuns possuem dois (*dual-core*), quatro (*quad-core*) ou até seis (*hexa-core*) núcleos de processamento em um único processador.

Um programa paralelo é executado por vários processadores ou núcleos de processamento, de modo que em cada um deles são executados um ou mais fluxos de controle, denominados *processos* ou *threads*. Um processo pode possuir várias threads que compartilham um mesmo espaço de endereçamento, enquanto cada processo possui um espaço de endereçamento próprio. A escolha do termo a ser empregado em geral depende da organização física da memória do ambiente de execução. Em sistemas com *memória distribuída*, em que cada processador ou núcleo tem acesso a uma memória própria privada, é comum empregar o termo processo. Por outro lado, o termo thread é usualmente utilizado para sistemas de *memória compartilhada*. Em ambientes desse tipo, uma memória compartilhada global armazena os dados de um programa e pode ser acessada por todos os processadores ou núcleos do sistema de hardware.

A criação de processos e threads pode ser feita através de um conceito

chamado *fork-join*. Em geral, programas paralelos executando em um sistema de memória compartilhada iniciam com uma única thread ativa, denominada *thread mestre*, responsável por executar a parte sequencial do código. Nas partes do código em que a computação paralela é necessária, chamadas *seções paralelas*, a thread existente cria threads filhas em uma operação chamada *fork*. Então, as threads filhas, juntamente com a thread mestre, executam em paralelo o código da seção paralela, ao final da qual o controle da execução retorna para a thread mestre, em uma operação chamada *join*.

Uma das primeiras etapas do projeto de um programa paralelo é a decomposição dos cálculos do programa sequencial em várias partes, chamadas *tarefas*. O tamanho de uma tarefa, dado por exemplo, em termos do número de instruções, é chamado *granularidade*. A execução de uma tarefa é atribuída a uma única thread ou processo (*escalonamento de tarefas*), que são então atribuídos a unidades de computação físicas (*escalonamento de threads ou processos*). O escalonamento de tarefas é responsável não apenas por atribuir tarefas a threads ou processos como também por definir a ordem em que essas tarefas serão processadas.

Tarefas diferentes podem ser executadas em paralelo por múltiplas threads ou processos. No entanto, as *dependências entre tarefas* são restrições importantes que devem ser consideradas no escalonamento destas para garantir a corretude de um programa paralelo. Por exemplo, se uma tarefa `T1` utiliza dados produzidos por uma outra tarefa `T2`, a execução de `T1` só pode ser iniciada após o término de `T2`. O principal objetivo do escalonamento de tarefas é conseguir uma boa divisão da carga de trabalho entre as threads ou processos, chamada *balanceamento de carga*. O ideal é conseguir atribuir às threads ou processos tarefas que realizam um mesmo número de cálculos e consequentemente demandam um mesmo tempo de processamento.

Uma outra questão importante que deve ser considerada no projeto de programas paralelos é a *sincronização* de threads e processos. Os métodos de sincronização em geral estão relacionados à forma em que informações são trocadas entre processos e threads, e esta depende da organização física da memória do ambiente de execução. Em ambientes de memória distribuída, informações são trocadas através da *passagem de mensagens* entre processos. Em ambientes de memória compartilhada, a troca de informações entre threads é feita através de *variáveis compartilhadas*, escritas por uma thread e lidas por outra thread.

A sincronização entre threads é responsável por coordenar o acesso de várias threads a dados compartilhados. Uma forma de coordenação se dá pelo uso de *barreiras de sincronização*. Nesse caso, quando uma thread executa as tarefas que lhe foram atribuídas e atinge um ponto do código marcado por uma barreira de

sincronização, ela deve esperar até que todas as outras threads executem completamente a parte do programa anterior à barreira. Apenas quando todas as threads atingem a barreira de sincronização, elas podem continuar executando a parte do código posterior à barreira.

Quando variáveis compartilhadas são utilizadas, em geral deve ser proibido o acesso simultâneo de múltiplas threads a uma mesma variável compartilhada, para a realização de operações concorrentes de leitura e escrita, uma vez que tal comportamento pode resultar em *condições de corrida*. Nesse contexto, o termo condição de corrida se refere ao fato do resultado de uma computação paralela poder ser dependente da ordem em que as threads acessam uma variável compartilhada. A presença de condições de corrida pode resultar em um comportamento não determinístico de um programa paralelo, devendo portanto ser evitada.

Se uma mesma operação deve ser aplicada de maneira independente a diferentes elementos de uma estrutura de dados, podemos utilizar a computação paralela para reduzir o tempo total de processamento. Nesse caso, os elementos da estrutura de dados podem ser distribuídos uniformemente entre os núcleos de processamento e cada núcleo executa a operação sobre os dados que lhe foram atribuídos. Esse tipo de paralelismo é chamado de *paralelismo de dados*. Em geral, linguagens de programação que fornecem paralelismo de dados funcionam de maneira similar a linguagens de programação sequenciais, em que um fluxo de controle único é utilizado. No entanto, essas linguagens possuem construções especiais que permitem o paralelismo de dados em estruturas de dados como arranjos. Esse modelo de computação paralela também é conhecido como *Single-Instruction, Multiple-Data* (*SIMD*).

O paralelismo de dados também pode ser explorado em modelos *Multiple-Instruction, Multiple-Data* (*MIMD*), nos quais núcleos de processamento têm acesso simultâneo a diferentes instruções que manipulam diferentes dados. Para tanto, em geral o modelo *Single Program Multiple Data* (*SPMD*) é utilizado, no qual um mesmo programa paralelo é executado de forma independente por todos os núcleos de processamento em paralelo. Nesse modelo, o paralelismo de dados é obtido quando cada núcleo fica responsável por executar as operações sobre uma parte específica dos dados.

Muitos algoritmos têm seus cálculos realizados através de um processo que percorre de forma iterativa uma estrutura de dados, chamado laço. Em geral, as iterações de um laço são executadas sequencialmente, isto é, o processamento da iteração i só é iniciado após o término do processamento da iteração $i - 1$. No entanto, se não existirem dependências entre as iterações de um laço, estas podem

ser executadas em uma ordem arbitrária ou até mesmo em paralelo por diferentes núcleos de processamento. Esses laços são chamados *laços paralelos*.

As duas principais medidas empregadas para avaliar o desempenho de programas paralelos são o *speedup* e a *eficiência paralela*. Assumindo que $t(p)$ denota o tempo de processamento necessário para executar um programa em p núcleos de processamento e que $t^*(1)$ denota o tempo de processamento necessário para executar a melhor implementação sequencial para o programa conhecida, o speedup $s(p)$ de um programa paralelo é definido como $s(p) := \frac{t^*(1)}{t(p)}$, isto é, o speedup indica quantas vezes o programa paralelo que usa p núcleos é mais rápido que a melhor implementação sequencial conhecida. Já a eficiência paralela $e(p)$ é uma medida do quão bem o programa paralelo utiliza o tempo nos p núcleos de processamento, sendo definida como $e(p) := \frac{s(p)}{p}$.

5.3.2 Detalhes de Implementação

O algoritmo paralelo do DSM foi implementado utilizando o padrão `OpenMP`®[OpenMP API, 2012], que possui uma Interface de Programação de Aplicação (*Application Programming Interface*, API) com uma coleção portável de rotinas, diretivas de compilação e variáveis de ambiente para a programação paralela em sistemas de memória compartilhada. Duas razões principais motivaram a nossa escolha pelo `OpenMP`. A primeira delas é a facilidade oferecida por sua API para a extensão da linguagem de programação `C++`, utilizada em todas as implementações deste trabalho, para o suporte a construções de paralelismo de dados, SPMD, balanceamento de carga, criação e destruição de threads e sincronização. A outra razão é o próprio modelo de programação do `OpenMP`, baseado em uma coleção de threads executando simultaneamente em múltiplos núcleos de processamento com acesso a uma memória compartilhada, exatamente a arquitetura que temos disponível para a realização dos nossos experimentos computacionais.

No `OpenMP`, a criação e a destruição de threads são feitas implicitamente de acordo com o padrão *fork-join*. Seções paralelas podem ser definidas através de diretivas de compilação. No início de uma seção paralela, a thread mestre realiza de maneira implícita uma operação *fork*, criando threads adicionais que executam em paralelo tarefas similares ou diferentes definidas no código da seção paralela. Ao final da seção paralela existe uma barreira de sincronização implícita, a partir da qual apenas a thread mestre continua sua execução (operação *join* implícita).

Para conseguir níveis razoáveis de paralelismo em nossa implementação paralela do DSM em um modelo de memória compartilhada, utilizamos estruturas de

dados apropriadas para representar os vértices, as arestas e os arcos de uma instância do PAGMGM. A escolha dessas estruturas foi feita com o intuito de conseguirmos principalmente um bom nível de paralelismo de dados. A seguir, listamos as principais informações armazenadas em cada uma dessas estruturas de dados:

- Informações armazenadas na estrutura de dados que representa o *vértice* $i \in V$:
  - (i) multiplicadores Lagrangeanos α_i e β_i,
  - (ii) custo Lagrangeano $\overline{u}_i$,
  - (iii) variável $\overline{y}_i$,
  - (iv) componentes s_{α_i} e s_{β_i} do vetor subgradiente s,
  - (v) componentes b_{α_i} e b_{β_i} do vetor de direção de busca b,
  - (vi) apontadores para as arestas pertencentes a $\delta(i)$ e
  - (vii) apontadores para os arcos pertencentes a $\delta^+(i)$ e $\delta^-(i)$.
- Informações armazenadas na estrutura de dados que representa a *aresta* $\{i, j\} \in E$:
  - (i) multiplicadores Lagrangeanos $\{\gamma_{ij}^r : r \in V\}$ e ω_{ij},
  - (ii) custo c_{ij},
  - (iii) custo Lagrangeano $\overline{c}_{ij}$,
  - (iv) variável $\overline{z}_{ij}$,
  - (v) componentes $\{s_{\gamma_{ij}^r} : r \in V\}$ e $s_{\omega_{ij}}$ do vetor subgradiente s e
  - (vi) componentes $\{b_{\gamma_{ij}^r} : r \in V\}$ e $b_{\omega_{ij}}$ do vetor de direção de busca b.
- Informações armazenadas na estrutura de dados que representa o *arco* $(i, j) \in A$:
  - (i) multiplicadores Lagrangeanos $\{\mu_{ij}^r : r \in V\}$,
  - (ii) custos Lagrangeanos $\{\overline{l}_{ij}^r : r \in V\}$,
  - (iii) variável $\{\overline{x}_{ij}^r : r \in V\}$,
  - (iv) componentes $\{s_{\mu_{ij}^r} : r \in V\}$ do vetor subgradiente s,
  - (v) componentes $\{b_{\mu_{ij}^r} : r \in V\}$ do vetor de direção de busca b,
  - (vi) apontador para a aresta $\{i, j\}$ que deu origem ao arco (i, j) e

(vii) apontador para o arco reverso (j, i).

Com o uso das estruturas de dados listadas, cada operação principal do DSM foi paralelizada utilizando uma mesma estrutura de três laços `for` aninhados: um para cada vértice $i \in V$, outro para cada arco $(i, j) \in \delta^+(i)$ e o último para cada vértice $r \in V$. Assim, o laço mais externo percorre a lista de vértices, o segundo laço processa a lista de arcos que partem de cada vértice (e consequentemente a lista de arestas que incidem em cada vértice) e o laço mais interno percorre a lista de raízes.

Para exemplificar o uso dos três laços `for` aninhados nas operações principais do DSM, mostramos na Figura 5.1 o pseudo-código da função COMPUTE-LAGRANGIAN-COSTS, implementada para calcular os custos Lagrangeanos associados às variáveis $\{\overline{z}_{ij} : \{i, j\} \in E\}$, $\{\overline{y}_i : i \in V\}$ e $\{\overline{x}^r_{ij} : r \in V,\ (i, j) \in A\}$. Optamos por manter no pseudo-código o nome da função no idioma inglês, da forma como foi escrito originalmente na nossa implementação em `C++` do DSM, mostrada na Figura 5.2.

```
COMPUTE-LAGRANGIAN-COSTS(V, E, A, k)
 1  for i ∈ V
 2      ū_i^k = (1 − d)α_i^k + (n − 2)β_i^k
 3      for (i, j) ∈ δ+(i)
 4          // seja {i, j} a aresta que deu origem ao arco (i, j)
 5          ū_i^k = ū_i^k + ω_ij^k
 6          if i < j // garante que a componente c̄_ij^k é calculada em
                     // uma única iteração do for mais externo (linha 1)
 7              c̄_ij^k = c_ij − (α_i^k + α_j^k) + (β_i^k + β_j^k) + ω_ij^k
 8          for r ∈ V
 9              l̄_ij^{r,k} = −γ_ij^{r,k}
10              if i < j
11                  c̄_ij^k = c̄_ij^k + γ_ij^{r,k}
12              if r ≠ i
13                  ū_i^k = ū_i^k + μ_ij^{r,k}
14                  l̄_ij^{r,k} = l̄_ij^{r,k} + μ_ij^{r,k}
15  return (c̄, ū, l̄)
```

Figura 5.1. Pseudo-código da função que calcula os custos Lagrangeanos na k-ésima iteração do DSM

Observe que empregando as estruturas de dados descritas, é possível calcular os custos Lagrangeanos $\{\overline{c}^k_{ij} : \{i, j\} \in E\}$, $\{\overline{u}^k_i : i \in V\}$ e $\{\overline{l}^{r,k}_{ij} : r \in V,\ (i, j) \in A\}$ em uma única execução dos três laços `for` aninhados. A mesma observação é válida

para o cálculo dos subgradientes, vetores de direção de busca, tamanho do passo e multiplicadores Lagrangeanos, cujos pseudo-códigos não são apresentados neste texto. Note também que quaisquer duas iterações distintas do laço `for` da linha 1 não acessam simultaneamente uma mesma variável compartilhada, ou seja, não existe nenhuma componente dos vetores de custos Lagrangeanos $\overline{c}$, $\overline{u}$ e $\overline{l}$ que tem seu valor calculado por duas ou mais iterações distintas do laço `for` mais externo. Portanto, esse laço pode ser executado em paralelo.

Assumindo que p denota o número de threads disponíveis, o laço `for` mais externo (linha 1 do código da Figura 5.1) é particionado em p blocos disjuntos com aproximadamente o mesmo número de iterações consecutivas. As instruções de cada bloco de iterações são então atribuídas a uma thread específica. Mais precisamente, o laço `for` mais externo possui n iterações. As primeiras $n \mod p$ threads ficam responsáveis pela execução de $\lceil n/p \rceil$ iterações e as outras threads ficam responsáveis por $\lfloor n/p \rfloor$ iterações. Assim, cada thread executa uma mesma operação em diferentes blocos de dados, evitando problemas como as condições de corrida. Na nossa implementação, empregamos a diretiva de compilação `parallel for` fornecida pela API do `OpenMP` para realizar o particionamento dos dados, o escalonamento de tarefas e a criação, sincronização e destruição de threads de maneira implícita.

Com o intuito de ilustrar o uso da API do `OpenMP` para a especificação de um laço paralelo, apresentamos na Figura 5.2 a implementação em `C++` da função `compute_lagrangian_costs`, que calcula em paralelo os custos Lagrangeanos em uma iteração do DSM. Assuma que essa função é um membro da classe `DeflectedSubgradientMethod`, assim como os arranjos `list_nodes`, `list_edges` e `list_arcs` e os parâmetros n e d.

Observe que a única diferença entre as implementações sequencial e paralela da função para o cálculo dos custos Lagrangeanos é a linha `#pragma omp parallel for private(node1, node2, edge, arc)`. O termo `parallel for` indica que o primeiro laço `for` que aparece após o termo é um laço paralelo. Já a expressão `private(node1, node2, edge, arc)` indica que uma cópia local privada das variáveis `node1`, `node2`, `edge` e `arc` deve ser feita para cada thread. O número de threads que são criadas é definido pela cláusula `omp_set_num_threads(p)`, chamada no início do programa. Utilizamos em nossos experimentos $p = 6$ threads.

Na nossa implementação paralela do DSM, também resolvemos os n subproblemas (5.16) em paralelo. Apesar das resoluções dos subproblemas (5.14) e (5.15) também serem paralelizáveis, optamos por utilizar algoritmos sequenciais, uma vez

```
void DeflectedSubgradientMethod::compute_lagrangian_costs()
{
   Node * node1, * node2;
   Edge * edge;
   Arc  * arc;
   int  i, j, r;

   #pragma omp parallel for private(node1, node2, edge, arc)
   for ( i = 0; i < n; ++i ) {
      node1 = list_nodes[i];
      node1->u_lagrangian_cost = ( 1 - d ) * node1->alpha +
                                 ( n - 2 ) * node1->beta;

      for ( size_t ind = 0; ind < node1->out_arcs.size(); ++ind ) {
         arc = node1->out_arcs[ind];
         edge = arc->original_edge;
         node2 = arc->destination;
         j = node2->id;

         node1->u_lagrangian_cost += edge->omega;

         if ( i < j ) edge->c_lagrangian_cost = edge->c + edge->omega -
                                                node1->alpha - node2->alpha +
                                                node1->beta + node2->beta;

         for ( r = 0; r < n; ++r ) {
            arc->l_lagrangian_cost[r] = -edge->gamma[r];
            if ( i < j ) edge->c_lagrangian_cost += edge->gamma[r];
            if ( r != i ) {
               node1->u_lagrangian_cost_ += arc->mu[r];
               arc->l_lagrangian_cost[r] += arc->mu[r];
            }
         }
      }
   }
}
```

Figura 5.2. Implementação em C++ da função que calcula os custos Lagrangeanos em uma iteração do DSM

que o tempo de computação necessário para resolvê-los representa uma fração insignificante do tempo total de processamento do DSM, principalmente para as instâncias de maior porte ($n \geq 100$).

Programas paralelos baseados no paradigma de paralelismo de dados podem apresentar uma baixa eficiência paralela, principalmente se as instâncias do problema consideradas não forem *regulares*. Nesse caso, surgem dificuldades adicionais no processo de balanceamento de carga. Para ilustrar tais dificuldades, considere que o cálculo dos custos $\overline{c}_{ij}^k$ e $\overline{c}_{pq}^k$ são realizados em paralelo por duas threads distintas. Se o número de multiplicadores Lagrangeanos que modificam o custo $\overline{c}_{ij}^k$ for consideravelmente diferente do número de multiplicadores avaliados no cálculo de $\overline{c}_{pq}^k$, uma thread tende a terminar sua tarefa muito antes da outra.

Como os passos básicos (1)-(7) do laço de repetição do DSM são inerentemente sequenciais, existe a necessidade de imposição de mecanismos de sincroniza-

ção de threads, para evitar acessos simultâneos a variáveis compartilhadas, impedir condições de corrida e garantir que o resultado do programa paralelo seja o mesmo fornecido pelo programa sequencial correspondente. Tais mecanismos devem garantir, por exemplo, que nenhuma thread inicie a resolução dos subproblemas (5.14)-(5.16) antes que todas as threads tenham finalizado seus laços para calcular os custos Lagrangeanos. Como uma consequência dessa sincronização, realizada implicitamente pelo `OpenMP` através de barreiras de sincronização, se uma thread executar o seu laço em um tempo muito menor que as demais, ela deve ficar esperando o término de todas as outras threads para poder começar a executar outras tarefas. Isso aumenta o tempo de espera das threads e reduz a eficiência paralela do programa, devendo portanto ser evitado.

Isso posto, é importante destacarmos que todas as instâncias do PAGMGM na literatura são definidas em grafos completos. Isso significa que, por exemplo, o número de multiplicadores Lagrangeanos que devem ser considerados no cálculo dos custos Lagrangeanos $\overline{c}_{ij}^{k}$ e $\overline{c}_{pq}^{k}$ são sempre iguais. De modo similar, o número de entradas $\overline{l}_{ij}^{r,k}$ que devem ser comparadas na resolução de dois subproblemas (5.16) diferentes também são iguais. O mesmo comportamento é observado em todos os demais passos do DSM. Portanto, as cargas de trabalho atribuídas às threads em seções paralelas do código tendem a ser uniformes, isto é, todas as threads tendem a gastar aproximadamente o mesmo tempo computacional para a execução de tarefas em paralelo. Por essa razão, a nossa implementação paralela do DSM obteve níveis de eficiência paralela bastante razoáveis.

5.4 Uma Heurística Lagrangeana

Com o objetivo de encontrar soluções viáveis de boa qualidade para o PAGMGM, implementamos uma Heurística Lagrangeana (HL) baseada em informações duais (custos Lagrangeanos). A heurística proposta envolve duas fases, uma construtiva e outra de melhoria. Na primeira fase, implementamos uma variação da heurística construtiva probabilística multipartida introduzida na Seção 4.1, que considera os custos Lagrangeanos obtidos nas iterações do DSM para determinar os conjuntos de vértices centrais. Na fase de melhoria, empregamos o algoritmo Local Branching proposto na Seção 4.3, formulando a restrição de Local Branching (4.5) a partir da melhor solução viável para o PAGMGM encontrada na fase construtiva.

Diferentemente de grande parte das heurísticas Lagrangeanas introduzidas na literatura, a heurística proposta neste trabalho não visa encontrar soluções viáveis

para o PAGMGM durante a execução do método empregado para resolver o problema Dual Lagrangeano (5.17). Na nossa implementação, a HL é chamada apenas quando o DSM é finalizado. A principal razão que justifica essa escolha é que, ao contrário da maioria das implementações de métodos de subgradientes, não utilizamos o custo de uma solução viável para o PAGMGM na definição do limite superior UB empregado no cálculo do tamanho do passo. Como dito anteriormente, devido à dualização de $O(n^3)$ restrições de igualdade (5.6), o número de entradas não nulas nos vetores de direção de busca tende a ser alto, fazendo com que o tamanho do passo assuma valores pequenos. Dessa forma, observamos que melhores limites inferiores Lagrangeanos são obtidos pelo DSM quando um limite superior trivial (dado pelo custo da árvore geradora de máximo custo de G, valorado com custos $\{c_{ij} : \{i,j\} \in E\}$ associados às suas arestas) é utilizado para determinar o valor de UB. Essa abordagem se mostrou efetiva para manter o tamanho do passo em uma ordem de magnitude razoável durante mais iterações do DSM, e foi utilizada nos experimentos computacionais relatados neste estudo.

A primeira fase da HL funciona de maneira similar à heurística CMP proposta no Capítulo 4 para gerar uma solução primal para o algoritmo Branch-and-cut. No entanto, custos Lagrangeanos são utilizados para guiar a escolha dos conjuntos de vértices centrais, ao invés de os selecionarmos de forma probabilística. Para tanto, durante a execução do DSM, armazenamos os custos Lagrangeanos $\{\overline{c}^k_{ij} : \{i,j\} \in E\}$ e $\{\overline{u}^k_i : i \in V\}$ associados às iterações $k = \{5, 10, 15, 20, \dots\}$ e a todas as iterações do DSM em que um melhor limite inferior para (5.17) é obtido. Assuma que $\mathcal{K}$ denote o conjunto dos índices associados a essas iterações do DSM. Para cada $k \in \mathcal{K}$, geramos uma solução viável para o PAGMGM, através da escolha de quatro conjuntos, a saber: um conjunto de vértices folha L^k, um conjunto de vértices centrais $C^k = V \setminus L^k$, um conjunto de arestas $E^k_{T_C} \subseteq E(C^k)$ conectando C^k e um conjunto de arestas $E^k_L \subseteq \delta(C^k)$ conectando os vértices folha L^k aos vértices centrais C^k.

Mais precisamente, primeiro computamos uma árvore geradora de custo mínimo T de G, valorado com custos Lagrangeanos $\{\overline{c}^k_{ij} : \{i,j\} \in E\}$ associados às suas arestas. De maneira análoga à notação empregada anteriormente neste texto, considere que $\{g^k_T(j) : j \in V\}$ denota o grau do vértice j em T. Em seguida, adicionamos a L^k os $n - \left\lfloor \frac{n-2}{d-1} \right\rfloor$ vértices de V com os menores valores $g^k_T(j)$, com empates resolvidos arbitrariamente. Os outros $\left\lfloor \frac{n-2}{d-1} \right\rfloor$ vértices $\{i : i \in V \setminus L^k\}$ são adicionados a C^k se $\overline{u}^k_i \geq 0$. Caso contrário, eles também são inseridos em L^k. Uma vez definidos os conjuntos L^k e C^k, a escolha das arestas de uma árvore geradora viável para o PAGMGM, isto é, das arestas dos conjuntos $E^k_{T_C}$ e E^k_L, é realizada através da fase (ii) da CMP. Para tanto, os custos Lagrangeanos $\{\overline{c}^k_{ij} : \{i,j\} \in E(C^k)\}$ são considerados

pelo algoritmo modificado de Kruskal para determinar as arestas $E^k_{T_C}$ que conectam os vértices centrais C^k. Para determinar as arestas E^k_L que conectam os vértices folha aos centrais, resolvemos o Programa Linear (4.1) considerando os custos originais $\{c_{ij} : \{i, j\} \in \delta(C^k)\}$.

Com o objetivo de reduzir o tempo de processamento necessário para executar por $|\mathcal{K}|$ vezes o procedimento descrito acima, exploramos novamente o paralelismo de dados. Para tanto, atribuímos a cada thread disponível a tarefa de gerar soluções viáveis para o PAGMGM associadas a um diferente subconjunto de índices de $\mathcal{K}$.

Após o término da primeira fase da HL, utilizamos a melhor solução viável encontrada para o PAGMGM como solução inicial da fase de melhoria da heurística. Nessa etapa, empregamos o algoritmo Local Branching apresentado no Capítulo 4. Para tanto, formulamos a restrição de Local Branching (4.5) a partir da melhor solução viável para o PAGMGM encontrada na fase construtiva da HL. Os parâmetros de configuração do método utilizados nos experimentos computacionais da Seção 4.3 foram aqui também empregados, com exceção daqueles relacionados ao limite de tempo global e ao limite de tempo para a resolução de cada nó do arcabouço externo de ramificação do método. Diferentemente da primeira fase, que é executada em paralelo, a segunda fase é executada sequencialmente.

5.5 Resultados Computacionais

Nesta seção, descrevemos os resultados computacionais obtidos pela Relaxação Lagrangeana, e os comparamos aos resultados fornecidos pelos métodos BB-MTZ, BC e LB.

Na apresentação dos resultados, utilizamos a sigla de Relaxação Lagrangeana, RL, para referenciar o método que envolve o algoritmo DSM seguido da HL. O mesmo limite de tempo de 6 horas utilizado para avaliar os métodos BB-MTZ, BC e LB foi empregado nos testes com a RL. Para tanto, definimos um limite de tempo de 14400 segundos (4 horas) para o algoritmo DSM. Ao final de sua execução, o tempo restante para completar 6 horas foi utilizado como limite de tempo da HL.

A RL foi avaliada no mesmo ambiente computacional utilizado nos experimentos com os métodos BB-MTZ, BC e LB. No entanto, utilizamos, em paralelo, os 6 núcleos de processamento disponíveis na máquina de testes para avaliar o método DSM,. Desse modo, os resultados da RL relatados nesta seção correspondem àqueles obtidos pela execução paralela do DSM em 6 núcleos de processamento.

Nos resultados apresentados a seguir, são consideradas as 36 instâncias que

não foram resolvidas na otimalidade por nenhum dos algoritmos anteriormente avaliados (BB-MTZ, BC e LB). Para cada instância, definimos $\underline{w}^*$ e $\overline{w}^*$ respectivamente como os melhores limites inferior e superior conhecidos, considerando os resultados obtidos pelos métodos BB-MTZ, LB, BC e RL. Desse modo, observe que $\overline{w}^*$ pode assumir valores diferentes dos utilizados no Capítulo 4, já que aqui consideramos também os resultados obtidos pela RL.

A Tabela 5.1 sumariza os resultados fornecidos pelos quatro métodos para cada tamanho de instância. As duas primeiras colunas indicam o valor de n e o número de instâncias em aberto com n vértices. Os resultados dos métodos BB-MTZ, BC, LB e RL são apresentados, nessa ordem, nas colunas seguintes. Para cada método, são indicados: o número de melhores limites inferiores obtidos pelo método iguais aos melhores limites inferiores conhecidos ($\underline{w} = \underline{w}^*$) e o número de melhores limites superiores obtidos pelo método iguais aos melhores limites superiores conhecidos ($\overline{w} = \overline{w}^*$).

Tabela 5.1. Síntese dos resultados obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], BC, LB e RL, para as instâncias cuja otimalidade não foi provada. Todos os métodos foram testados no mesmo ambiente computacional.

		BB-MTZ		BC		LB		RL	
n	#instâncias	$\underline{w} = \underline{w}^*$	$\overline{w} = \overline{w}^*$	$\underline{w} = \underline{w}^*$	$\overline{w} = \overline{w}^*$	$\underline{w} = \underline{w}^*$	$\overline{w} = \overline{w}^*$	$\underline{w} = \underline{w}^*$	$\overline{w} = \overline{w}^*$
100	6	0	0	6	0	0	6	0	4
200	6	0	0	0	0	0	1	6	5
300	6	0	0	0	0	0	1	6	5
400	9	1	2	6	1	0	1	2	5
500	9	1	2	8	3	0	3	0	1

Considerando todos os métodos avaliados nesta dissertação e as 36 instâncias em aberto, podemos observar pela Tabela 5.1 que, de um modo geral, os melhores limites inferiores foram obtidos pelos métodos BC e RL, enquanto que os melhores limites superiores foram dados pelo LB e pela RL.

Mais precisamente, para 14 das 36 instâncias consideradas, a RL foi capaz de prover, em 4 horas de processamento (limite de tempo do DSM), melhores limites inferiores que aqueles fornecidos pelo algoritmo BC, obtidos após a exploração de milhares de nós em sua árvore de enumeração durante 6 horas. Em particular, para todas as instâncias com 200 e 300 vértices, os melhores limites inferiores conhecidos foram obtidos pela RL. Observe que, em 10 dessas 12 instâncias, a RL também forneceu os melhores limites superiores conhecidos. No total, 16 novos melhores limites superiores para o PAGMGM foram obtidos pela RL.

No entanto, os limites inferiores obtidos pela RL para as instâncias com 400 e 500 vértices foram piores que os melhores limites fornecidos pelo método BB-MTZ

ou pelo algoritmo BC em praticamente todos os casos. Esses resultados são justificados pelo fato de que, devido ao tamanho dessas instâncias, o custo computacional de cada iteração do DSM é alto, mesmo considerando a sua implementação paralela, e por isso o número de iterações realizadas pelo método durante as 4 horas de processamento é muito baixo. Desse modo, são obtidas aproximações ruins dos multiplicadores Lagrangeanos ótimos. Nos resultados detalhados apresentados a seguir, essas questões são evidenciadas.

5.5.1 Resultados Detalhados

Nesta seção, apresentamos os resultados detalhados dos experimentos realizados com a RL, e os comparamos com os resultados obtidos pelo algoritmo de planos de corte proposto neste trabalho para avaliar $w(P_I)$ e com os melhores resultados obtidos pelos métodos BB-MTZ, BC e LB para a resolução do PAGMGM.

Inicialmente, apresentamos uma comparação entre os melhores limites inferiores obtidos pela RL e pelo algoritmo de planos de corte introduzido na Seção 3.5.1 para avaliar o limite de PL da formulação P_I, para as instâncias das classes *CRD* e *SYM*. Não consideramos nessa análise o procedimento iterativo proposto em [Gouveia & Telhada, 2011], uma vez que os tempos computacionais gastos por esse método foram superiores aos tempos gastos pelo procedimento que considera todas as n raízes possíveis desde o início (veja Tabela 3.4).

A Tabela 5.2 mostra essa comparação. A primeira coluna indica os ids das instâncias consideradas. As duas colunas seguintes mostram, respectivamente, o valor de $w(P_I)$ avaliado pelo método de planos de corte e o tempo computacional gasto para avaliá-lo ($t_{w(P_I)}$, em segundos). As duas colunas seguintes apresentam o melhor limite inferior obtido pela RL ($\underline{w}$) e o tempo computacional gasto para obtê-lo ($t_{\underline{w}}$, em segundos), respectivamente. Por fim, as duas últimas colunas mostram as razões entre os melhores limites inferiores obtidos por cada método $\left(\frac{\underline{w}}{w(P_I)}\right)$ e os tempos computacionais gastos para obtê-los $\left(\frac{t_{\underline{w}}}{t_{w(P_I)}}\right)$. É importante destacar que para 12 instâncias reportadas na tabela (ids 22 a 30 e 52 a 54), o método de planos de corte não foi capaz de avaliar $w(P_I)$ durante 6 horas de processamento. Desse modo, os valores correspondentes mostrados na tabela, prefixados com o símbolo '$\star$', indicam o limite inferior para $w(P_I)$ fornecido pelo algoritmo de planos de corte após 6 horas de execução.

Como pode ser observado na tabela, a RL foi capaz de obter boas aproximações dos limites de PL da formulação P_I, em tempos computacionais muito inferiores aos gastos pelo método de planos de corte. Limitando a análise às 42 instâncias para as

quais $w(P_I)$ pôde ser avaliado por meio do algoritmo de planos de corte no limite de tempo de 6 horas, o *gap* percentual médio entre os melhores limites inferiores obtidos pela RL e os limites de PL da formulação P_I foi de 0.25%. Para esses casos, o tempo total gasto pela RL para obter tais limites foi duas ordens de grandeza inferior ao tempo correspondente gasto pelo algoritmo de planos de corte. Observe ainda que, para 9 instâncias consideradas, o melhor limite inferior dado pela RL foi exatamente $w(P_I)$.

Ao analisarmos as 12 instâncias cujos limites de PL da formulação P_I não puderam ser avaliados pelo método de planos de corte, observamos que os melhores limites inferiores dados pela RL foram, em média, $2{,}90\%$ mais fortes que os fornecidos pelo método de planos de corte, tendo sido obtidos em $0{,}96\%$ do tempo computacional em que o algoritmo de planos de corte foi executado. Considerando apenas as instâncias com 100 vértices, observamos uma diferença ainda maior entre os limites fornecidos pelos dois métodos. Nesses casos, em média, os melhores limites inferiores obtidos pela RL foram $5{,}34\%$ mais fortes que os fornecidos pelo método de planos de corte.

A Tabela 5.3 apresenta resultados detalhados dos limites inferiores obtidos pelo método DSM para as instâncias cujos certificados de otimalidade não são conhecidos. Na primeira coluna, são apresentados os ids das instâncias. Nas duas colunas seguintes, são indicados, respectivamente, o melhor limite inferior conhecido para cada instância (considerando os resultados dos métodos BB-MTZ, LB e BC) e o algoritmo que o obteve. Os resultados do método DSM são apresentados nas seis colunas subsequentes, a saber: número de iterações executadas (#iterações), valor do parâmetro ε ao final de sua execução, melhor limite inferior obtido ($\underline{w}$), tempo computacional considerando a execução em 6 núcleos de processamento ($t(6)$, em segundos), *speedup* ($sp(6)$) e eficiência paralela ($e(6)$).

Como pode ser observado pela tabela, para todas as instâncias com 300 ou menos vértices, o DSM foi finalizado após realizar o número máximo de iterações estabelecido (20000) ou após ε assumir um valor muito pequeno, menor que 10^{-4}. Para todas as instâncias de 200 e 300 vértices, os melhores limites inferiores obtidos pelo DSM foram mais fortes que aqueles fornecidos pelo BC após 6 horas de execução. Cabe ressaltar que tais limites foram obtidos em tempos computacionais consideravelmente inferiores ao tempo total em que o BC foi executado. Considerando separadamente as instâncias de 200 e 300 vértices, o tempo total gasto pelo DSM para obter os seus melhores limites superiores correspondeu a $12{,}32\%$ e $54{,}29\%$ do tempo total em que o BC foi executado para cada tamanho dessas instâncias, respectivamente. Em contrapartida, para as instâncias de 100 vértices, os limites duais

dados pelo DSM foram mais fracos que os melhores conhecidos até o momento. Analisando os resultados do BC apresentados anteriormente, é possível observar que, para as instâncias de 100 vértices, o método consegue explorar milhares de nós em sua árvore de enumeração e prover limites duais mais fortes que os fornecidos para as maiores instâncias. Essa é uma possível justificativa para o fato da RL não ter conseguido obter limites inferiores melhores que o BC para esses casos.

Ao analisarmos os resultados do DSM para as instâncias de 400 e 500 vértices, observamos que o método foi finalizado, em todos os casos, após atingir o limite de tempo imposto de 4 horas de processamento. Observe que, em apenas duas dessas instâncias (ids 73 e 74), os limites inferiores obtidos pelo DSM foram superiores aos melhores conhecidos até o momento. A razão que justifica esses resultados é o fato de que, para instâncias desse tamanho, o custo computacional de cada iteração do DSM é alto, devido principalmente à resolução dos subproblemas (5.16) por meio de um procedimento com complexidade computacional $O(n^3)$, e por isso o número de iterações realizadas pelo método durante o limite de tempo imposto é muito baixo. Note que, ao final da execução do DSM, os valores de ε (e consequentemente os tamanhos dos passos) para as instâncias de 500 vértices ainda são consideravelmente altos. Desse modo, o algoritmo DSM não consegue obter boas aproximações dos multiplicadores Lagrangeanos ótimos, o que implica na baixa qualidade dos limites inferiores fornecidos.

Para o cálculo do *speedup* e da eficiência paralela obtidos pela nossa implementação do DSM, executamos a versão sequencial do método durante o mesmo número de iterações executadas pela sua versão paralela. Os resultados mostraram que, sob uma perspectiva de programação paralela, níveis razoáveis de eficiência foram alcançados. Acreditamos que tais resultados são devidos principalmente à maneira como implementamos os laços paralelos e ao fato de todas as instâncias testadas serem definidas em grafos completos, permitindo assim um bom balanceamento de carga. Cabe ressaltar que os valores observados para o *speedup* aumentam à medida que o valor de n cresce, o que sugere que, para instâncias maiores, uma melhor granularidade foi obtida para as tarefas paralelas, reduzindo assim o *overhead* de comunicação e sincronização.

A Tabela 5.4 apresenta resultados detalhados dos limites superiores obtidos pela HL para as instâncias em aberto. Na primeira coluna, são apresentados os ids das instâncias. Nas duas colunas seguintes, são indicados, respectivamente, o melhor limite superior conhecido para cada instância (considerando os resultados dos métodos BB-MTZ, LB e BC) e o algoritmo que o obteve. Os resultados da HL são apresentados nas cinco colunas subsequentes, a saber: o melhor limite superior

obtido na primeira fase da HL ($\overline{w}_{f1}$), o melhor limite superior obtido pela HL após a execução de sua segunda fase ($\overline{w}$), os tempos computacionais gastos para executar a primeira (t_{f1}, em segundos) e a segunda (t_{f2}, em segundos) fases da HL e o tempo total gasto pela RL (t, em segundos), que corresponde ao somatório dos tempos gastos pelo DSM e pela HL.

A tabela mostra que para 19 das 27 instâncias com 400 ou menos vértices, a HL foi capaz de encontrar uma solução viável de custo igual ou inferior ao custo da melhor solução até então conhecida. Mais precisamente, 15 novos melhores limites superiores foram obtidos para esses casos. Tais limites foram, em média, $1{,}60\%$ mais fortes que os melhores limites já obtidos pelos demais métodos. No entanto, pode-se observar que os limites primais obtidos para as instâncias com 500 vértices foram consideravelmente piores que os melhores limites conhecidos ($10{,}13\%$, em média). Isso pode ser justificado pelo fato dos limites primais obtidos pela HL serem fortemente dependentes dos melhores limites inferiores obtidos pelo DSM, como pode ser observado pelas tabelas. Em 11 dos 14 casos em que o DSM forneceu um limite inferior mais forte que o melhor conhecido, o melhor limite superior obtido pela HL também foi mais forte que o melhor limite já obtido. Além disso, nos outros 3 casos, o custo da melhor solução viável encontrada pela HL foi, em média, apenas $0{,}37\%$ pior que o custo da melhor solução conhecida.

Apesar do DSM não ter fornecido limites duais mais fortes que os melhores conhecidos para as instâncias de 100 vértices, podemos notar pela Tabela 5.4 que ainda assim os limites superiores obtidos pela HL foram iguais aos melhores conhecidos para 4 das 6 instâncias desse tamanho. A razão que justifica esse resultado é o tempo total disponível para a execução da segunda fase da HL para esses casos. Observe que, devido aos baixos tempos computacionais gastos para executar o DSM e a primeira fase da HL para as instâncias de 100 vértices, o algoritmo Local Branching empregado na segunda fase da HL pôde ser executado por quase 6 horas, conseguindo assim encontrar soluções viáveis de boa qualidade. No entanto, ao analisarmos as instâncias de 500 vértices, para as quais a segunda fase da HL foi executada por menos de 2 horas, observamos que em 8 dos 9 casos a solução dada pela primeira fase da HL não foi aprimorada até o final da execução do método. No único caso em que uma solução aprimorante foi encontrada na segunda fase da HL (instância de id 89), um novo melhor limite superior para o PAGMGM foi obtido.

Tabela 5.2. Comparação entre o método de planos de corte proposto neste estudo para avaliar $w(P_I)$ e a RL, para as instâncias das classes *CRD* e *SYM*.

	Planos de corte		RL			
id	$w(P_I)$	$t_{w(P_I)}$	$\underline{w}$	$t_{\underline{w}}$	$\frac{\underline{w}}{w(P_I)}$	$\frac{t_{\underline{w}}}{t_{w(P_I)}}$
1	3964,9	23,6	3960,3	3,0	0,9989	0,1267
2	3632,5	6,4	3632,5	2,8	1,0000	0,4401
3	4166,5	3,7	4166,5	2,3	1,0000	0,6081
4	4885,8	47,7	4878,1	3,5	0,9984	0,0743
5	4549,0	34,2	4544,9	3,7	0,9991	0,1093
6	5209,1	56,5	5201,7	4,2	0,9986	0,0736
7	5435,7	243,9	5430,6	13,4	0,9991	0,0548
8	5696,1	390,8	5685,7	14,7	0,9982	0,0377
9	5540,7	436,8	5534,6	17,0	0,9989	0,0388
10	6571,7	1349,1	6551,8	19,3	0,9970	0,0143
11	7027,2	2054,6	7003,3	17,3	0,9966	0,0084
12	6996,0	1893,1	6972,4	17,0	0,9966	0,0090
13	9326,1	6298,6	9296,7	20,9	0,9968	0,0033
14	9557,8	4979,4	9540,9	19,7	0,9982	0,0039
15	9643,4	4932,4	9624,9	19,4	0,9981	0,0039
16	6377,2	2869,7	6364,3	75,4	0,9980	0,0263
17	6420,5	2836,2	6406,3	70,8	0,9978	0,0249
18	6807,3	2985,8	6795,6	67,4	0,9983	0,0226
19	7753,3	16114,6	7718,6	83,6	0,9955	0,0052
20	7708,9	15341,5	7683,5	85,6	0,9967	0,0056
21	8153,2	8898,2	8111,8	88,9	0,9949	0,0100
22	*10961,4	21600,0	10999,9	94,8	1,0035	0,0044
23	*10956,5	21600,0	11000,0	94,7	1,0040	0,0044
24	*11687,6	21600,0	11765,6	95,0	1,0067	0,0044
25	*8275,3	21600,0	8768,3	329,1	1,0596	0,0152
26	*8636,9	21600,0	9158,6	319,3	1,0604	0,0148
27	*8555,7	21600,0	9111,2	324,9	1,0649	0,0150
28	*12107,9	21600,0	12665,3	332,9	1,0460	0,0154
29	*12194,1	21600,0	12773,3	332,9	1,0475	0,0154
30	*12489,3	21600,0	13013,2	333,1	1,0420	0,0154
31	1194,0	6,0	1194,0	2,9	1,0000	0,4760
32	1435,0	2,6	1435,0	2,2	1,0000	0,8599
33	1408,0	4,1	1408,0	2,2	1,0000	0,5392
34	1765,0	35,8	1754,0	3,7	0,9938	0,1035
35	2013,5	31,4	2006,8	3,7	0,9967	0,1176
36	2008,0	22,3	2008,0	2,7	1,0000	0,1231
37	1274,7	182,3	1274,1	13,6	0,9996	0,0747
38	1176,5	275,4	1174,8	14,4	0,9986	0,0522
39	1597,5	60,2	1597,5	11,5	1,0000	0,1909
40	1954,3	1106,0	1937,1	14,9	0,9912	0,0135
41	1741,8	788,8	1733,5	19,9	0,9952	0,0252
42	2442,0	964,9	2434,1	15,3	0,9968	0,0158
43	3812,8	2605,3	3789,9	16,7	0,9940	0,0064
44	3720,9	1672,5	3708,6	17,2	0,9967	0,0103
45	4521,3	3918,0	4496,6	16,1	0,9945	0,0041
46	1359,0	981,8	1359,0	62,1	1,0000	0,0633
47	1433,3	1665,0	1432,5	75,2	0,9994	0,0451
48	1521,0	865,7	1521,0	54,6	1,0000	0,0631
49	1974,7	13062,2	1954,2	72,0	0,9896	0,0055
50	2126,4	7031,0	2119,1	74,1	0,9965	0,0105
51	2170,6	8820,3	2162,4	88,5	0,9962	0,0100
52	*4273,3	21600,0	4342,9	80,7	1,0163	0,0037
53	*4207,4	21600,0	4192,4	78,6	0,9964	0,0036
54	*4311,7	21600,0	4315,4	77,5	1,0009	0,0036

Tabela 5.3. Resultados computacionais dos limites inferiores obtidos pela RL, para as instâncias cuja otimalidade não foi provada. As colunas associadas aos melhores limites inferiores conhecidos correspondem aos limites obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], LB e BC.

	Melhor conhecido		RL (DSM)					
id	$\underline{w}$	algoritmo	#iterações	ε	$\underline{w}$	$t(6)$	$sp(6)$	$e(6)$
25	8839,3	BC	20000	0,00	8768,3	329,1	2,52	0,4196
26	9303,6	BC	19580	0,00	9158,6	319,3	2,53	0,4211
27	9228,3	BC	20000	0,00	9111,2	324,9	2,53	0,4214
55	5142,6	BC	19574	0,00	5060,2	320,2	2,52	0,4198
56	4870,3	BC	18649	0,00	4869,2	307,0	2,53	0,4219
57	5210,2	BC	18446	0,00	5189,3	301,5	2,53	0,4218
61	6429,0	BC	20000	0,00	6541,3	2635,8	2,47	0,4122
62	6597,9	BC	20000	0,00	6691,0	2639,7	2,39	0,3983
63	6882,5	BC	20000	0,00	6991,2	2643,8	2,48	0,4131
64	9017,9	BC	20000	0,00	9041,3	2674,4	2,40	0,4006
65	9350,7	BC	20000	0,00	9399,2	2699,7	2,40	0,3995
66	9391,2	BC	20000	0,00	9394,5	2671,2	2,41	0,4019
67	8016,0	BC	20000	0,00	8202,9	11698,2	3,38	0,5633
68	7798,8	BC	20000	0,00	8002,8	11746,1	3,33	0,5549
69	8099,0	BC	20000	0,00	8292,5	11735,2	3,37	0,5611
70	11355,4	BC	20000	0,00	11385,2	11697,0	3,29	0,5488
71	10936,2	BC	20000	0,00	10981,4	11718,0	3,30	0,5507
72	11416,1	BC	20000	0,00	11522,9	11771,2	3,31	0,5518
73	9121,2	BC	8096	0,07	9169,4	14400,0	3,89	0,6491
74	9227,9	BC	8087	0,07	9250,3	14400,0	3,88	0,6467
75	8945,5	BC	8078	0,09	8936,0	14400,0	3,90	0,6497
76	12976,0	BC	8015	0,05	12814,9	14400,0	3,85	0,6409
77	12876,1	BC	8001	0,03	12775,9	14400,0	3,84	0,6407
78	12601,9	BC	8017	0,05	12408,1	14400,0	3,82	0,6367
79	18932,9	BB-MTZ	7940	0,04	18525,4	14400,0	3,84	0,6397
80	18712,8	BC	7939	0,02	18519,8	14400,0	3,87	0,6457
81	18708,4	BC	7974	0,02	18457,9	14400,0	3,86	0,6432
82	9868,9	BC	3749	0,42	9126,3	14400,0	4,08	0,6804
83	10077,0	BC	3743	0,52	9111,1	14400,0	4,05	0,6756
84	9904,7	BC	3752	0,27	9453,5	14400,0	4,09	0,6822
85	13736,4	BC	3723	0,21	12548,0	14400,0	4,07	0,6784
86	14258,5	BC	3718	0,27	12674,3	14400,0	4,06	0,6771
87	14071,7	BC	3708	0,27	12466,8	14400,0	4,05	0,6749
88	19851,2	BC	3685	0,27	16333,3	14400,0	4,07	0,6778
89	20419,4	BC	3692	0,27	16828,7	14400,0	4,07	0,6783
90	20395,8	BB-MTZ	3686	0,27	16751,8	14400,0	4,11	0,6850

Tabela 5.4. Resultados computacionais dos limites superiores obtidos pela RL, para as instâncias cuja otimalidade não foi provada. As colunas associadas aos melhores limites superiores conhecidos correspondem aos limites obtidos pelos métodos HVNS [Martins & Souza, 2009], BB-MTZ [Akgún & Tansel, 2010], LB e BC.

	Melhor conhecido		RL (HL)				
id	$\overline{w}$	algoritmo	$\overline{w}_{f1}$	$\overline{w}$	t_{f1}	t_{f2}	t
25	9251	LB	9861	9251	6,8	21264,1	21600
26	9557	LB	10093	9557	6,1	21274,6	21600
27	9622	LB	10041	9635	5,7	21269,4	21600
55	5317	LB	5534	5317	6,6	21273,2	21600
56	5022	LB	5241	5022	6,0	21287,0	21600
57	5428	LB	5683	5429	5,7	21292,8	21600
61	7075	BC	7752	6964	29,1	18935,1	21600
62	7118	LB	8002	7129	29,7	18930,6	21600
63	7492	BC	8111	7407	30,2	18926,0	21600
64	9652	BC	10263	9552	19,8	18905,8	21600
65	9801	LB	10651	9752	19,8	18880,5	21600
66	9904	LB	10561	9894	18,8	18910,0	21600
67	9195	LB	10048	9088	88,3	9813,5	21600
68	8877	LB	9675	8728	87,0	9766,9	21600
69	9296	LB	10087	9051	86,9	9777,9	21600
70	12381	LB	13698	12418	52,7	9850,3	21600
71	12246	BC	13370	12019	52,3	9829,8	21600
72	12725	LB	13276	12378	54,0	9774,8	21600
73	10670	LB	11842	10560	112,3	7087,7	21600
74	10791	BB-MTZ	11959	10863	106,9	7093,1	21600
75	10336	BB-MTZ	11760	10532	110,4	7089,6	21600
76	14611	BC	16635	14702	43,5	7156,5	21600
77	14679	LB	15762	14451	44,8	7155,2	21600
78	14184	LB	15739	14279	44,2	7155,8	21600
79	20345	LB	21653	20154	27,3	7172,7	21600
80	20473	BC	21326	19677	28,3	7171,7	21600
81	20427	LB	20927	19969	27,4	7172,6	21600
82	11872	BB-MTZ	13224	13224	136,4	7063,6	21600
83	12512	BB-MTZ	13629	13629	134,9	7065,1	21600
84	12035	BC	13182	13182	150,5	7049,5	21600
85	15617	BC	18138	18138	43,1	7156,9	21600
86	16325	LB	18913	18913	42,5	7157,5	21600
87	16466	BC	18599	18599	43,1	7156,9	21600
88	23465	LB	24879	24879	23,1	7176,9	21600
89	23668	LB	25411	22823	21,9	7178,1	21600
90	22726	LB	25891	25891	22,8	7177,2	21600

5.6 Comentários

Pela nossa avaliação, a Relaxação Lagrangeana proposta neste capítulo apresentou resultados promissores para o PAGMGM. O método conseguiu fornecer 14 novos melhores limites inferiores para o problema, por meio de uma implementação paralela do *Deflected Subgradient Method* (DSM) [Camerini et al., 1975] para resolver o Problema Dual Lagrangeano. Em particular, os limites inferiores obtidos pelo método para todas as instâncias avaliadas com 200 e 300 vértices foram mais fortes que os melhores limites inferiores conhecidos. Pelo lado primal, a Heurística Lagrangeana proposta também obteve bons resultados, fornecendo 16 novos melhores limites superiores para o PAGMGM.

Devido aos razoáveis níveis de eficiência paralela obtidos pela nossa implementação do DSM, acreditamos que, para as instâncias maiores consideradas ($n \in \{400, 500\}$), uma implementação paralela do método projetada para sistemas de memória distribuída e compartilhada (um ambiente com vários processadores *multi-core*) tem potencial para melhorar esses resultados. Os experimentos realizados sugerem que, ao realizar um número suficientemente grande de iterações do DSM para as instâncias dessas dimensões, resultados melhores tanto pelo lado dual como pelo lado primal devem ser obtidos.

Capítulo 6

Conclusões e Trabalhos Futuros

Nesta dissertação, estudamos o *Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo* (PAGMGM), recentemente proposto por Almeida et al. [2006] e demonstrado ser NP-Difícil para os casos $d \geq 4$ em [Almeida et al., 2006] e $d = 3$ em [Almeida et al., 2010]. Novas formulações de Programação Inteira foram aqui introduzidas, assim como algoritmos de otimização heurísticos e exatos. Os métodos propostos foram avaliados computacionalmente para todas as 90 instâncias utilizadas na literatura do PAGMGM, e os seus resultados foram comparados àqueles obtidos pelos melhores métodos já empregados para a resolução do problema, a saber: as meta-heurísticas introduzidas em [Martins & Souza, 2009] e o algoritmo Branch-and-bound apresentado em [Akgún & Tansel, 2010] (BB-MTZ). Com os métodos propostos neste estudo, novos certificados de otimalidade e novos melhores limites inferiores e superiores foram fornecidos para várias instâncias do PAGMGM.

As formulações introduzidas foram comparadas em relação aos seus limites de Programação Linear (PL) sob uma perspectiva teórica e computacional. Duas dessas formulações, uma direcionada e outra não direcionada, são baseadas em um número exponencial de restrições de eliminação de subcircuitos. Resultados teóricos e experimentais apresentados demonstraram que a formulação direcionada domina a não direcionada em relação aos seus limites de PL. Motivados pelo fato da formulação direcionada não ser simétrica em relação aos limites de PL, introduzimos uma reformulação compacta e simétrica para o PAGMGM, obtida com a aplicação da técnica de Reformulação por Interseção proposta por Gouveia & Telhada [2008]. Essa reformulação apresentou os limites de PL mais fortes conhecidos para o problema.

Embora a reformulação por interseção forneça melhores limites inferiores que

a formulação direcionada, na prática os tempos computacionais gastos para avaliá-los torna proibitiva sua aplicação direta em procedimentos de enumeração do tipo Branch-and-bound, a menos que algum tratamento algorítmico adicional seja empregado. Desse modo, introduzimos dois métodos para a resolução exata do PAGMGM baseados na formulação direcionada.

O primeiro método proposto foi um algoritmo Branch-and-cut (BC), inicializado a partir de uma solução primal para o PAGMGM, obtida por meio de uma heurística construtiva multipartida probabilística proposta nesta dissertação. Os experimentos computacionais mostraram que o algoritmo BC claramente domina o método BB-MTZ em termos de tempos computacionais. O método forneceu 6 novos certificados de otimalidade para o PAGMGM, assim como 20 novos melhores limites inferiores e 4 novos melhores limites superiores, considerando as 36 instâncias ainda em aberto na literatura.

Com o objetivo de tentar obter melhores soluções viáveis para o problema em menores tempos computacionais, implementamos um segundo método exato para o PAGMGM: um algoritmo Local Branching (LB), que emprega o próprio BC como resolvedor interno. O LB foi capaz de resolver na otimalidade as mesmas instâncias que o método BC, tendo encontrado as soluções ótimas para esses casos em tempos inferiores aos gastos pelo BC para encontrá-las. Em contrapartida, em geral o LB gastou maiores tempos computacionais para provar a otimalidade dessas soluções. O método forneceu 12 novos melhores limites superiores para o PAGMGM, mas, como esperado, obteve limites duais mais fracos que o BC para todas as instâncias.

Dada a dificuldade em se resolver as maiores instâncias do PAGMGM por meio dos métodos BC e LB, baseados na formulação direcionada, propusemos uma Relaxação Lagrangeana (RL) baseada na reformulação por interseção. Com o objetivo de resolver mais rapidamente o Problema Dual Lagrangeano, implementamos uma versão paralela de uma variação do Método de Subgradiente. Pelo lado primal, desenvolvemos uma Heurística Lagrangeana baseada no algoritmo LB. Pela nossa avaliação, os resultados obtidos pela RL foram bastante promissores. Em particular, 14 novos melhores limites inferiores e 20 novos melhores limites superiores foram fornecidos para o PAGMGM.

Em uma avaliação geral, os métodos propostos nesta dissertação (BC, LB e RL) conseguiram melhorar os resultados da literatura do PAGMGM em vários aspectos, mas ainda não conseguem resolver, em tempos computacionais razoáveis, as maiores instâncias do problema (com 200 ou mais vértices). Diferentes abordagens podem ser seguidas com o objetivo de aprimorar tais métodos.

Considerando o algoritmo BC, primeiramente gostaríamos de mencionar que

os resultados apresentados neste trabalho são melhores que aqueles obtidos pelas versões do método publicadas em [Martinez & Cunha, 2010, 2011]. Essa melhoria se deve principalmente à política de seleção de cortes que são incluídos nos problemas relaxados em cada iteração do método de planos de corte. Algumas possibilidades a serem investigadas em trabalhos futuros para aprimorar o BC são: (i) a separação das restrições direcionadas de eliminação de subcircuitos em $O(n^3)$, por meio do algoritmo proposto em [Hao & Orlin, 1994], em detrimento do procedimento de separação utilizado neste trabalho, cuja complexidade computacional de tempo é $O(n^4)$, (ii) o desenvolvimento de procedimentos capazes de separar em tempos computacionais razoáveis as novas desigualdades válidas para o PAGMGM propostas neste trabalho (veja os Teoremas 1 e 2 e o Corolário 1) e (iii) o uso de outras desigualdades válidas obtidas a partir de um estudo poliedral que pretendemos desenvolver para o PAGMGM.

Como o método LB emprega o algoritmo BC para resolver os subproblemas definidos pelo seu arcabouço externo de ramificação, qualquer aprimoramento deste implica diretamente em uma versão melhorada do método LB. Pretendemos ainda fornecer uma implementação paralela do LB em uma continuação deste trabalho.

Acreditamos que uma implementação paralela da RL, projetada para sistemas de memória distribuída e compartilhada (um ambiente com vários processadores *multi-core*), tem grande potencial para melhorar seus resultados aqui relatados. Essa tese é reforçada ao compararmos os resultados apresentados àqueles obtidos nos experimentos computacionais realizados anteriormente com o método, relatados em um trabalho submetido recentemente para publicação [Martinez & Cunha, 2012]. Nesse estudo, foram utilizados 4 núcleos de processamento, ao passo que nos experimentos descritos nesta dissertação utilizamos um processador *hexa-core*. Apesar de observarmos uma piora na eficiência paralela à medida que mais núcleos são empregados, a implementação do método executada em 6 núcleos foi capaz de realizar um número consideravelmente maior de iterações do Método de Subgradiente no limite de tempo imposto e, por isso, conseguiu fornecer melhores limites inferiores para as maiores instâncias consideradas.

Outras direções para trabalhos futuros que pretendemos seguir são: (i) o uso da Relaxação Lagrangeana para fixar variáveis em soluções ótimas do PAGMGM, e a consequente utilização desse método em uma etapa de pré-processamento do algoritmo Branch-and-cut e (ii) o desenvolvimento de um algoritmo Branch-and-bound para a resolução exata do PAGMGM, baseado na Relaxação Lagrangeana.

Referências Bibliográficas

Ahuja, R. K.; Magnanti, T. L. & Orlin, J. B. (1993). *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc.

Akgún, I. & Tansel, B. (2010). Min-degree constrained minimum spanning tree problem: New formulation via Miller-Tucker-Zemlin constraints. *Computers and Operations Research*, 37(1):72–82.

Almeida, A. M.; Martins, P. & Souza, M. C. (2006). Min-Degree Constrained Minimum Spanning Tree Problem: Complexity, proprieties and formulations. Relatório técnico 6/2006, Centro de Investigação Operacional, Universidade de Lisboa.

Almeida, A. M.; Martins, P. & Souza, M. C. (2010). md-MST is NP-hard for $d \geq 3$. *Electronic Notes in Discrete Mathematics*, 36:9–15.

Andrade, R.; Lucena, A. & Maculan, N. (2006). Using Lagrangian dual information to generate degree constrained spanning trees. *Discrete Applied Mathematics*, 154(5):703–717.

Applegate, D.; Bixby, R. E.; Chvatal, V. & Cook, W. (1995). Finding cuts in the TSP. *Technical Report 95-09, DIMACS*.

Bahiense, L.; Barahona, F. & Porto, O. (2003). Solving steiner tree problems in graphs with lagrangian relaxation. *Journal of Combinatorial Optimization*, 7(3):259–282.

Bahiense, L.; Maculan, N. & Sagastizábal, C. (2002). The volume algorithm revisited: relation with bundle methods. *Mathematical Programming*, 94:41–70.

Barahona, F. & Anbil, R. (2000). The volume algorithm: producing primal solutions with subgradient method. *Mathematical Programming*, 87:385–399.

Beasley, J. (1993). Lagrangean Relaxation. Em Reeves, C., editor, *Modern Heuristic Techniques*. Blackwell Scientific Press, Oxford.

Caccetta, L. & Hill, S. (2001). A Branch and cut Method for the Degree-Constrained Minimum Spanning Tree Problem. *Networks*, 37(2):74–83.

Camerini, P.; Fratta, L. & Maffioli, F. (1975). On improving relaxation methods by modified gradient techniques. *Mathematical Programming Study*, 3:26–34.

Chopra, S.; Gorres, E. R. & Rao, M. R. (1992). Solving the Steiner Tree Problems on a Graph Using Branch and Cut. *ORSA Journal on Computing*, 4(3):320–335.

Cunha, A. S. & Lucena, A. (2007). Lower and Upper Bounds for the Degree-Constrained Minimal Spanning Tree Problem. *Networks*, 50:55–66.

Current, J. R.; Revelle, C. S. & Cohon, J. L. (1986). The hierarchical network design problem. *European Journal of Operational Research*, 27:5–66.

Dantzig, G. B.; Fulkerson, D. R. & Johnson, S. M. (1954). Solution of a large scale traveling salesman problem. *Operations Research*, 2:393–410.

Deo, N. & Hakimi, S. (1968). The shortest generalized Hamiltonian tree. Em *Proceedings of the 6th Alberton Conference*, pp. 879–888.

Deo, N. & Kumar, N. (1997). Computation of Constrained Spanning Trees: A Unified Approach. *Network Optimization*, 450:194–220.

Desrochers, M. & Laporte, G. (1991). Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints. *Operations Research Letters*, 10:27–36.

Edmonds, J. (1967). Optimum branchings. *J. Research of the National Bureau of Standards*, 71B:233–240.

Edmonds, J. (1971). Matroids and the Greedy Algorithm. *Mathematical Programming*, 1:127–136.

Fischer, M. L. (1981). The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18.

Fischetti, M. & Lodi, A. (2003). Local Branching. *Mathematical Programming*, 98:23–47.

Fischetti, M.; Polo, C. & Scantamburlo, M. (2004). A local branching heuristic for mixed-integer programs with 2-level variables, with an application to a telecommunication network design problem. *Networks*, 44(2):61–72.

Garey, M. R. & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, New York.

Geoffrion, A. M. (1974). Lagrangian Relaxation for integer programming. *Mathematical Programming Study*, 2:82–114.

Gouveia, L. & Telhada, J. (2008). The multi-weighted Steiner Tree problem: A reformulation by intersection. *Computers and Operations Research*, 35:3599–3611.

Gouveia, L. & Telhada, J. (2011). Reformulation by intersection method on the MST problem with lower bound on the number of leaves. *Lecture Notes in Computer Science*, 6701:83–91.

Guignard, M. (2003). Lagrangian relaxation. *TOP - Sociedad de Estadística e Investigación Operativa*, 11(2):151–228.

Hao, J. & Orlin, J. B. (1994). A Faster Algorithm for Finding the Minimum Cut in a Directed Graph. *J. Algorithms*, 17:424–446.

Held, M. & Karp, R. (1970). The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18:1138–1162.

Held, M. & Karp, R. (1971). The Traveling Salesman Problem and Minimum Spanning Trees: Part II. *Mathematical Programming*, 1:6–25.

Held, M. H.; Wolfe, P. & Crowder, H. D. (1974). Validation of Subgradient Optimization. *Mathematical Programming*, 6:62–88.

IBM ILOG CPLEX Optimizer (2012). `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`. Visitado em 15 de janeiro de 2012.

ILOG Concert Technology (2012). `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/interfaces/#concert_technology`. Visitado em 15 de janeiro de 2012.

Junger, M.; Reinelt, G. & Thienel, S. (1995). Practical problem solving with cutting plane algorithms in combinatorial optimization. *Combinatorial Optimization, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pp. 111–152.

Karnaugh, M. (1976). A new class of algorithms for multipoint network optimization. *IEEE Transactions on Communication*, 24:5:500–505.

Kelley, J. E. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712.

Koch, T. & Martin, A. (1998). Solving Steiner Tree Problems in Graphs to Optimality. *Networks*, 32:207–232.

Krishnamoorthy, M.; Ernst, T. A. & Sharaiha, Y. M. (2001). Comparison of Algorithms for the Degree constrained minimum spanning tree. *Journal of Heuristics*, 7:587–611.

Kruskal, J. (1956). On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7:48–50.

Langevin, A.; Soumis, F. & Desrosiers, J. (1990). Classification of traveling salesman problem formulations. *Operations Research Letters*, 9:127–132.

Lemarechal, C. (1974). An algorithm for minimizing convex functions. Em *Proceedings of IFIP'74 Congress*, pp. 552–556. North Holland.

Lucena, A. & Beasley, J. (1996). *Branch and cut algorithms*, pp. 187–221. Oxford University Press, Inc.

Lucena, A. & Resende, M. G. C. (2004). Strong lower bounds for the prize-collecting Steiner problem in Graphs. *Discrete Applied Mathematics*, 141:277–294.

Maculan, N. (1987). The Steiner problem in graphs. *Annals of Discrete Mathematics*, 31:185–212.

Magnanti, T. & Wolsey, L. (1995). Optimal Trees. Em et al, O. B., editor, *Handbooks in OR and MS*, volume 7, pp. 503–615. North-Holland.

Martinez, L. C. & Cunha, A. S. (2009). Um arcabouço Local Branching para Problemas de Otimização Combinatória aplicado ao Problema da Árvore de Custo Mínimo com k arestas. Em *XLI Simposio Brasileiro de Pesquisa Operacional*.

Martinez, L. C. & Cunha, A. S. (2010). Finding min-degree constrained spanning trees faster with a branch-and-cut algorithm. *Electronic Notes in Discrete Mathematics*, 36:311–318. ISCO - International Symposium on Combinatorial Optimization.

Martinez, L. C. & Cunha, A. S. (2011). The Min-Degree Constrained Minimum Spanning Tree Problem: Formulations and Branch-and-cut algorithm. *Discrete Applied Mathematics*. In Press, doi:10.1016/j.dam.2011.08.008.

Martinez, L. C. & Cunha, A. S. (2012). A Parallel Lagrangian Relaxation Algorithm for the Min-Degree Constrained Minimum Spanning Tree Problem. Em *ISCO - International Symposium on Combinatorial Optimization*. Submetido para publicação.

Martins, P. (2007). Enhanced second order algorithm applied to the capacitated minimum spanning tree problem. *Computers and Operations Research*, 34:2495–2519.

Martins, P. & Souza, M. C. (2009). VNS and second order heuristics for the min-degree constrained minimum sp anning tree problem. *Computers and Operations Research*, 36:2669–2982.

Martín, I. R. & González, J. S. (2010). A local branching heuristic for the capacitated fixed-charge network design problem. *Computers and Operations Research*, 37(3):575–581.

Miller, C. E.; Tucker, A. W. & Zemlin, R. A. (1960). Integer programming formulations and travelling salesman problems. *Journal of the Association of Computing Machinery*, 7:326–329.

Mladenović, N. & Hansen, P. (1997). Variable Neighborhood Search. *Computers & Operations Research*, 24(11):1097–1100.

Narula, S. C. & Ho, C. A. (1980). Degree-constrained minimum spanning tree. *Computers and Operations Research*, 7:239–249.

OpenMP API (2012). `http://www.openmp.org`. Visitado em 15 de janeiro de 2012.

Padberg, M. & Sung, T.-Y. (1991). An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming*, 52:315–357.

Padberg, M. & Wolsey, L. (1983). Trees and cuts. *Annals of Discrete Mathematics*, 17:511–517.

Padberg, M. W. & Rinaldi, G. (1991). A Branch-and-Cut algorithm for resolution of large scale of Symmetric Traveling Salesman Problem. *SIAM Review*, 33:60–100.

Papadimitriou, C. H. (1994). *Computational Complexity*. Addinson-Wesley.

Pochet, Y. & Wolsey, L. (2006). *Production Planning by Mixed Integer Programming*. Springer.

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Tech. J.*, 36:1389–1401.

Rauber, T. & Rünger, G. (2010). *Parallel Programming: for Multicore and Cluster Systems*. Springer.

Rei, W.; Cordeau, J.; Gendreau, M. & Soriano, P. (2009). Accelerating benders decomposition by local branching. *INFORMS Journal on Computing*, 21(2):333–345.

Rei, W.; Gendreau, M. & Soriano, P. (2010). A hybrid monte carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science*, 44(1):136–146.

Ribeiro, C. C. & Souza, M. C. (2002). Variable Neighborhood Search for the Degree-Constrained Minimum Spanning Tree Problem. *Discrete Applied Mathematics*, 118:43–54.

Savelsbergh, M. & Volgenant, A. (1985). Edge exchanges in the degree constrained minimum spanning tree problem. *Computers and Operations Research*, 12(4):341–348.

Souza, M. C. & Martins, P. (2008). Skewed VNS Enclosing Second Order Algorithm for the Degree Constrained Minimum Spanning Tree Problem. *European Journal of Operational Research*, 191:677–690.

Valle, C. A.; Martinez, L. C.; Cunha, A. S. & Mateus, G. R. (2011). Heuristic and exact algorithms for a minmax selective vehicle routing problem. *Computers and Operations Research*, 38(7):1054–1065.

Volgenant, A. (1989). A Lagrangean approach to the degree-constrained minimum spanning tree problem. *European Journal of Operational Research*, 39:325–331.

West, D. B. (2001). *Introduction to Graph Theory*. Prentice Hall.

Wolsey, L. (1998). *Integer Programming*. John Wiley and Sons.

Apêndice A

Matriz de Custo de uma Instância de 10 Vértices para o PAGMGM

Apresentamos a seguir a matriz de custo M da instância de 10 vértices utilizada nas provas das Observações 1 e 2.

$$M = \begin{bmatrix} - & 19 & 80 & 29 & 75 & 52 & 51 & 43 & 96 & 57 \\ & - & 59 & 23 & 61 & 98 & 20 & 66 & 44 & 74 \\ & & - & 56 & 43 & 17 & 54 & 64 & 59 & 61 \\ & & & - & 48 & 96 & 73 & 43 & 37 & 45 \\ & & & & - & 95 & 7 & 76 & 75 & 81 \\ & & & & & - & 27 & 25 & 76 & 74 \\ & & & & & & - & 81 & 86 & 96 \\ & & & & & & & - & 94 & 84 \\ & & & & & & & & - & 67 \\ & & & & & & & & & - \end{bmatrix}$$

Apêndice B

Compilação dos Melhores Limites Inferiores e Superiores Conhecidos para o PAGMGM

Neste apêndice, são sumarizados os resultados relacionados aos melhores limites inferiores e superiores conhecidos para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau Mínimo (PAGMGM), considerando as 90 instâncias utilizadas na literatura do problema. Relatamos os melhores resultados obtidos com as meta-heurísticas propostas por Martins & Souza [2009] (HVNS), com o algoritmo Branch-and-bound proposto por Akgún & Tansel [2010] (BB-MTZ) e com os métodos Branch-and-cut (BC), Local Branching (LB) e Relaxação Lagrangeana (RL) propostos nesta dissertação. O objetivo deste apêndice é fornecer ao leitor uma visão geral dos resultados alcançados pelos métodos já aplicados na literatura (incluindo este trabalho) para a resolução do PAGMGM.

Cabe destacar que todos os resultados apresentados neste apêndice, com exceção dos fornecidos pelas meta-heurísticas HVNS, foram obtidos em experimentos computacionais realizados em um mesmo ambiente computacional (veja Seção 3.6.2), com o mesmo limite de tempo de 6 horas. Já os resultados apresentados para as heurísticas HVNS foram retirados dos experimentos relatados em [Martins & Souza, 2009]. Nesse estudo, o ambiente computacional e os limites de tempo utilizados foram diferentes.

As Tabelas B.1, B.2 e B.3 apresentam os limites inferiores e superiores obtidos por cada um destes métodos para as instâncias das classes *CRD*, *SYM* e *ALM*, respectivamente. Nas quatro primeiras colunas da tabela, são informados os dados das instâncias avaliadas, a saber: id, nome, o número de vértices da instância (n) e

Tabela B.1. Melhores limites inferiores obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], BC e RL e melhores limites superiores obtidos pelos métodos HVNS [Martins & Souza, 2009], BB-MTZ, BC, LB e RL, para as instâncias da classe *CRD*.

Instâncias CRD				Limites inferiores			Limites superiores				
id	nome	n	d	BB-MTZ	BC	RL	HVNS	BB-MTZ	BC	LB	RL
1	CRD30-1	30	3	**4026.0**	**4026.0**	3960.3	**4026**	**4026**	**4026**	**4026**	**4026**
2	CRD30-2	30	3	**3793.0**	**3793.0**	3632.4	**3793**	**3793**	**3793**	**3793**	**3793**
3	CRD30-3	30	3	**4293.0**	**4293.0**	4166.5	**4293**	**4293**	**4293**	**4293**	**4293**
4	CRD30-1	30	5	**5026.0**	**5026.0**	4878.1	**5026**	**5026**	**5026**	**5026**	**5026**
5	CRD30-2	30	5	**4648.0**	**4648.0**	4544.9	**4648**	**4648**	**4648**	**4648**	**4648**
6	CRD30-3	30	5	**5425.0**	**5425.0**	5201.7	**5425**	**5425**	**5425**	**5425**	**5425**
7	CRD50-1	50	3	**5512.0**	**5512.0**	5430.6	**5512**	**5512**	**5512**	**5512**	**5512**
8	CRD50-2	50	3	**5813.0**	**5813.0**	5685.7	**5813**	**5813**	**5813**	**5813**	**5813**
9	CRD50-3	50	3	**5590.0**	**5590.0**	5534.6	**5590**	**5590**	**5590**	**5590**	**5590**
10	CRD50-1	50	5	**6908.0**	**6908.0**	6551.8	**6908**	**6908**	**6908**	**6908**	**6908**
11	CRD50-2	50	5	**7204.0**	**7204.0**	7003.3	7238	**7204**	**7204**	**7204**	**7204**
12	CRD50-3	50	5	**7277.0**	**7277.0**	6972.4	**7277**	**7277**	**7277**	**7277**	**7277**
13	CRD50-1	50	10	**9633.0**	**9633.0**	9296.7	**9633**	**9633**	**9633**	**9633**	**9633**
14	CRD50-2	50	10	**9743.0**	**9743.0**	9540.9	**9743**	**9743**	**9743**	**9743**	**9743**
15	CRD50-3	50	10	**9855.0**	**9855.0**	9624.9	**9855**	**9855**	**9855**	**9855**	**9855**
16	CRD70-1	70	3	6267.4	**6516.0**	6364.2	6609	**6516**	**6516**	**6516**	**6516**
17	CRD70-2	70	3	6211.9	**6586.0**	6406.3	6621	6621	**6586**	**6586**	**6586**
18	CRD70-3	70	3	6569.7	**7053.0**	6795.6	7058	7066	**7053**	**7053**	**7053**
19	CRD70-1	70	5	7699.3	**8134.0**	7718.6	8177	**8134**	**8134**	**8134**	**8134**
20	CRD70-2	70	5	7726.5	**7943.0**	7683.5	7971	**7943**	**7943**	**7943**	**7943**
21	CRD70-3	70	5	8260.3	**8419.0**	8111.8	**8419**	8421	**8419**	**8419**	**8419**
22	CRD70-1	70	10	**11235.0**	**11235.0**	10999.9	11355	**11235**	**11235**	**11235**	**11235**
23	CRD70-2	70	10	**11373.0**	**11373.0**	11000.0	11395	**11373**	**11373**	**11373**	**11373**
24	CRD70-3	70	10	**11979.0**	**11979.0**	11765.6	11986	**11979**	**11979**	**11979**	**11979**
25	CRD100-1	100	5	8559.2	**8839.3**	8768.3	9387	9334	9349	**9251**	**9251**
26	CRD100-2	100	5	8964.1	**9303.6**	9158.6	9728	9562	9571	**9557**	**9557**
27	CRD100-3	100	5	8699.4	**9228.3**	9111.2	9739	9635	9653	**9622**	9635
28	CRD100-1	100	10	**12916.0**	**12916.0**	12665.3	13006	**12916**	**12916**	**12916**	**12916**
29	CRD100-2	100	10	**13026.0**	**13026.0**	12773.3	13255	**13026**	**13026**	**13026**	**13026**
30	CRD100-3	100	10	**13365.0**	**13365.0**	13013.2	**13365**	**13365**	**13365**	**13365**	**13365**

o valor do parâmetro de grau mínimo d. Os ids são inteiros que escolhemos para referenciar as instâncias do PAGMGM nesta dissertação. As três colunas seguintes indicam os melhores limites inferiores obtidos pelos métodos BB-MTZ, BC e RL, nessa ordem, ao final da execução do método ou ao final do tempo limite de execução. As últimas cinco colunas apresentam os melhores limites superiores fornecidos pelos métodos HVNS, BB-MTZ, BC, LB e RL, respectivamente. São destacados em negrito os melhores limites obtidos para cada instância.

A Tabela B.4 mostra uma comparação entre o número de melhores limites inferiores e superiores conhecidos para o PAGMGM obtidos pelos métodos HVNS, BB-MTZ, BC, LB e RL, para cada tamanho de grafo associado às 36 instâncias do problema para as quais nenhum método avaliado foi capaz de prover o certificado de otimalidade. As primeiras duas colunas mostram o número de vértices n e o

Tabela B.2. Melhores limites inferiores obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], BC e RL e melhores limites superiores obtidos pelos métodos HVNS [Martins & Souza, 2009], BB-MTZ, BC, LB e RL, para as instâncias da classe *SYM*.

Instâncias SYM				Limites inferiores			Limites superiores				
id	nome	n	d	BB-MTZ	BC	RL	HVNS	BB-MTZ	BC	LB	RL
31	SYM30-1	30	3	**1197.0**	**1197.0**	1194.0	**1197**	**1197**	**1197**	**1197**	**1197**
32	SYM30-2	30	3	**1435.0**	**1435.0**	**1435.0**	**1435**	**1435**	**1435**	**1435**	**1435**
33	SYM30-3	30	3	**1408.0**	**1408.0**	**1408.0**	**1408**	**1408**	**1408**	**1408**	**1408**
34	SYM30-1	30	5	**1765.0**	**1765.0**	1754.0	**1765**	**1765**	**1765**	**1765**	**1765**
35	SYM30-2	30	5	**2090.0**	**2090.0**	2006.8	**2090**	**2090**	**2090**	**2090**	**2090**
36	SYM30-3	30	5	**2008.0**	**2008.0**	**2008.0**	**2008**	**2008**	**2008**	**2008**	**2008**
37	SYM50-1	50	3	**1278.0**	**1278.0**	1274.1	1298	**1278**	**1278**	**1278**	**1278**
38	SYM50-2	50	3	**1178.0**	**1178.0**	1174.8	1188	**1178**	**1178**	**1178**	**1178**
39	SYM50-3	50	3	**1615.0**	**1615.0**	1597.5	1620	**1615**	**1615**	**1615**	**1615**
40	SYM50-1	50	5	**2054.0**	**2054.0**	1937.1	**2054**	**2054**	**2054**	**2054**	**2054**
41	SYM50-2	50	5	**1760.0**	**1760.0**	1733.5	**1760**	**1760**	**1760**	**1760**	**1760**
42	SYM50-3	50	5	**2525.0**	**2525.0**	2434.1	2541	**2525**	**2525**	**2525**	**2525**
43	SYM50-1	50	10	**4121.0**	**4121.0**	3789.9	**4121**	**4121**	**4121**	**4121**	**4121**
44	SYM50-2	50	10	**4166.0**	**4166.0**	3708.6	**4166**	**4166**	**4166**	**4166**	**4166**
45	SYM50-3	50	10	**4979.0**	**4979.0**	4496.6	**4979**	**4979**	**4979**	**4979**	**4979**
46	SYM70-1	70	3	**1360.0**	**1360.0**	1359.0	1362	**1360**	**1360**	**1360**	**1360**
47	SYM70-2	70	3	**1448.0**	**1448.0**	1432.5	1471	**1448**	**1448**	**1448**	1449
48	SYM70-3	70	3	**1521.0**	**1521.0**	**1521.0**	1551	**1521**	**1521**	**1521**	**1521**
49	SYM70-1	70	5	**2028.0**	**2028.0**	1954.2	2240	**2028**	**2028**	**2028**	**2028**
50	SYM70-2	70	5	**2165.0**	**2165.0**	2119.1	2496	**2165**	**2165**	**2165**	**2165**
51	SYM70-3	70	5	**2210.0**	**2210.0**	2162.3	2242	**2210**	**2210**	**2210**	**2210**
52	SYM70-1	70	10	**4979.0**	**4979.0**	4342.9	5055	**4979**	**4979**	**4979**	**4979**
53	SYM70-2	70	10	**4787.0**	**4787.0**	4192.4	4912	**4787**	**4787**	**4787**	**4787**
54	SYM70-3	70	10	**4997.0**	**4997.0**	4315.4	5098	**4997**	**4997**	**4997**	**4997**

número de instâncias com n vértices para as quais não são conhecidos os certificados de otimalidade. As quatro colunas seguintes indicam o número de melhores limites inferiores obtidos por cada um dos métodos BB-MTZ, BC, LB e RL, respectivamente. Por fim, as últimas cinco colunas apresentam o número de melhores limites superiores fornecidos pelos métodos HVNS, BB-MTZ, BC, LB e RL, nessa ordem.

A Tabela B.5 apresenta os melhores limites inferiores e superiores conhecidos para o PAGMGM, assim como os *gaps* de dualidade percentuais associados, para as 90 instâncias utilizadas na literatura do problema. Os resultados apresentados foram obtidos pelos métodos HVNS, BB-MTZ, BC, LB e RL. A tabela é divida em três blocos de trinta linhas e quatro colunas. Em cada bloco, a primeira coluna apresenta os ids das instâncias. As colunas seguintes indicam, respectivamente, o melhor limite inferior conhecido para a instância ($\underline{w}^*$), o melhor limite superior conhecido para a instância ($\overline{w}^*$) e o *gap* de dualidade, em valores percentuais, associado a esses melhores limites, definido como $100(\overline{w}^* - \underline{w}^*)/\overline{w}^*$. As entradas nas colunas *gap* (%) correspondentes às instâncias resolvidas na otimalidade mostram o símbolo "-".

Tabela B.3. Melhores limites inferiores obtidos pelos métodos BB-MTZ [Akgún & Tansel, 2010], BC e RL e melhores limites superiores obtidos pelos métodos HVNS [Martins & Souza, 2009], BB-MTZ, BC, LB e RL, para as instâncias da classe *ALM*.

Instâncias ALM				Limites inferiores			Limites superiores				
id	nome	n	d	BB-MTZ	BC	RL	HVNS	BB-MTZ	BC	LB	RL
55	ALM100-1	100	5	4831.2	**5142.6**	5060.1	5439	5457	5353	**5317**	**5317**
56	ALM100-2	100	5	4719.4	**4870.3**	4869.2	5207	5197	5055	**5022**	**5022**
57	ALM100-3	100	5	5058.5	**5210.1**	5189.2	5456	5450	5450	**5428**	5429
58	ALM100-1	100	10	**7164.0**	**7164.0**	7065.2	7180	**7164**	**7164**	**7164**	**7164**
59	ALM100-2	100	10	**6886.0**	**6886.0**	6726.2	6915	**6886**	**6886**	**6886**	**6886**
60	ALM100-3	100	10	**7382.0**	**7382.0**	7138.1	7509	**7382**	**7382**	**7382**	**7382**
61	ALM200-1	200	5	6183.4	6429.0	**6541.3**	7467	7217	7075	7122	**6964**
62	ALM200-2	200	5	6361.2	6597.9	**6691.0**	7680	7180	7245	**7118**	7129
63	ALM200-3	200	5	6626.4	6882.5	**6991.2**	8217	7556	7492	7514	**7407**
64	ALM200-1	200	10	8919.8	9017.9	**9041.3**	10391	9680	9652	9764	**9552**
65	ALM200-2	200	10	9263.0	9350.7	**9399.2**	10238	9916	9820	9801	**9752**
66	ALM200-3	200	10	9278.3	9391.2	**9394.5**	10533	10039	9978	9904	**9894**
67	ALM300-1	300	5	7780.2	8016.0	**8202.9**	9871	9211	9705	9195	**9088**
68	ALM300-2	300	5	7564.4	7798.8	**8002.8**	10532	9295	9407	8877	**8728**
69	ALM300-3	300	5	7870.4	8099.0	**8292.5**	10887	9359	9458	9296	**9051**
70	ALM300-1	300	10	11280.5	11355.4	**11385.2**	13899	12603	12891	**12381**	12418
71	ALM300-2	300	10	10826.1	10936.2	**10981.4**	13210	12723	12246	12272	**12019**
72	ALM300-3	300	10	11279.2	11416.0	**11522.9**	13792	13063	13110	12725	**12378**
73	ALM400-1	400	5	8837.9	9121.2	**9169.4**	12487	10917	11079	10670	**10560**
74	ALM400-2	400	5	8967.4	9227.9	**9250.2**	13877	**10791**	11036	11076	10863
75	ALM400-3	400	5	8630.6	**8945.5**	8936.0	12379	**10336**	10711	10447	10532
76	ALM400-1	400	10	12927.9	**12976.0**	12814.9	17309	15055	**14611**	14705	14702
77	ALM400-2	400	10	12779.1	**12876.1**	12775.9	16595	14774	15086	14679	**14451**
78	ALM400-3	400	10	12403.5	**12601.9**	12408.1	16439	15142	14506	**14184**	14279
79	ALM400-1	400	20	**18932.9**	18929.0	18525.4	21339	23287	21260	20345	**20154**
80	ALM400-2	400	20	18675.7	**18712.8**	18519.8	21299	23951	20473	20520	**19677**
81	ALM400-3	400	20	18667.5	**18708.4**	18457.9	22049	22445	21478	20427	**19969**
82	ALM500-1	500	5	9608.3	**9868.9**	9126.3	14626	**11872**	11881	13328	13224
83	ALM500-2	500	5	9756.4	**10077.0**	9111.1	14039	**12512**	12518	12990	13629
84	ALM500-3	500	5	9543.0	**9904.7**	9453.5	13521	12190	**12035**	12562	13182
85	ALM500-1	500	10	13627.0	**13736.4**	12548.0	19342	17900	**15617**	16187	18138
86	ALM500-2	500	10	14131.0	**14258.5**	12674.3	18138	33992	16344	**16325**	18913
87	ALM500-3	500	10	13972.8	**14071.7**	12466.8	18269	17168	**16466**	18644	18599
88	ALM500-1	500	20	19839.6	**19851.2**	16333.3	24999	42194	24508	**23465**	24879
89	ALM500-2	500	20	20396.4	**20419.4**	16828.7	24823	23871	24900	23668	**22823**
90	ALM500-3	500	20	**20395.8**	20384.0	16751.8	25468	53380	23901	**22726**	25891

Tabela B.4. Número de melhores limites inferiores e superiores conhecidos para o PAGMGM obtidos por cada um dos métodos HVNS [Martins & Souza, 2009], BB-MTZ [Akgún & Tansel, 2010], BC, LB e RL, para as instâncias do problema cujos certificados de otimalidade não foram fornecidos.

	#instâncias	Limites inferiores				Limites superiores				
n	em aberto	BB-MTZ	BC	LB	RL	HVNS	BB-MTZ	BC	LB	RL
100	6	0	6	0	0	0	0	0	6	4
200	6	0	0	0	6	0	0	0	1	5
300	6	0	0	0	6	0	0	0	1	5
400	9	1	6	0	2	0	2	1	1	5
500	9	1	8	0	0	0	2	3	3	1

Tabela B.5. Melhores limites inferiores e superiores conhecidos para o PAGMGM, e os *gaps* de dualidade associados, para todas as instâncias das classes *CRD*, *SYM* e *ALM* já utilizadas na literatura do problema.

Instâncias CRD				Instâncias SYM e ALM				Instâncias ALM			
id	$\underline{w}^*$	$\overline{w}^*$	*gap* (%)	id	$\underline{w}^*$	$\overline{w}^*$	*gap* (%)	id	$\underline{w}^*$	$\overline{w}^*$	*gap* (%)
1	4026	4026	-	31	1197	1197	-	61	6542	6964	6.06
2	3793	3793	-	32	1435	1435	-	62	6692	7118	5.98
3	4293	4293	-	33	1408	1408	-	63	6992	7407	5.60
4	5026	5026	-	34	1765	1765	-	64	9042	9552	5.34
5	4648	4648	-	35	2090	2090	-	65	9400	9752	3.61
6	5425	5425	-	36	2008	2008	-	66	9395	9894	5.04
7	5512	5512	-	37	1278	1278	-	67	8203	9088	9.74
8	5813	5813	-	38	1178	1178	-	68	8003	8728	8.31
9	5590	5590	-	39	1615	1615	-	69	8293	9051	8.37
10	6908	6908	-	40	2054	2054	-	70	11386	12381	8.04
11	7204	7204	-	41	1760	1760	-	71	10982	12019	8.63
12	7277	7277	-	42	2525	2525	-	72	11523	12378	6.91
13	9633	9633	-	43	4121	4121	-	73	9170	10560	13.16
14	9743	9743	-	44	4166	4166	-	74	9251	10791	14.27
15	9855	9855	-	45	4979	4979	-	75	8946	10336	13.45
16	6516	6516	-	46	1360	1360	-	76	12976	14611	11.19
17	6586	6586	-	47	1448	1448	-	77	12877	14451	10.89
18	7053	7053	-	48	1521	1521	-	78	12602	14184	11.15
19	8134	8134	-	49	2028	2028	-	79	18933	20154	6.06
20	7943	7943	-	50	2165	2165	-	80	18713	19677	4.90
21	8419	8419	-	51	2210	2210	-	81	18709	19969	6.31
22	11235	11235	-	52	4979	4979	-	82	9869	11872	16.87
23	11373	11373	-	53	4787	4787	-	83	10077	12512	19.46
24	11979	11979	-	54	4997	4997	-	84	9905	12035	17.70
25	8840	9251	4.44	55	5143	5317	3.27	85	13737	15617	12.04
26	9304	9557	2.65	56	4871	5022	3.01	86	14259	16325	12.66
27	9229	9622	4.08	57	5211	5428	4.00	87	14072	16466	14.54
28	12916	12916	-	58	7164	7164	-	88	19852	23465	15.40
29	13026	13026	-	59	6886	6886	-	89	20420	22823	10.53
30	13365	13365	-	60	7382	7382	-	90	20396	22726	10.25