ÁRVORES ÓTIMAS EM GRAFOS: MODELOS, ALGORITMOS E APLICAÇÕES

Alexandre Salles da Cunha

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Abilio Pereira de Lucena Filho, Ph.D.

Prof. Nelson Maculan Filho, D.Sc.

Prof. Cid Carvalho de Souza, Ph.D.

Prof. Geraldo Robson Mateus, D.Sc.

Prof. Felipe Maia Galvão França, Ph.D.

Prof. Adilson Elias Xavier, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

MAIO DE 2006

CUNHA, ALEXANDRE SALLES DA

Árvores ótimas em grafos: modelos, algoritmos e aplicações [Rio de Janeiro] 2006

XVI, 199 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2006)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Árvores

2 - Algoritmos Relax-and-Cut

3 - Algoritmos Branch-and-Cut

4 - Atalhos Lagrangeanos para algoritmos de Planos de Corte

5 - Otimização Combinatória

I. COPPE/UFRJ II. Título (série)

Aos meus pais

Ao meu amigo e orientador Prof. Abilio Lucena, muito obrigado pelo empenho e pelas portas que me ajudou a abrir. Se um dia eu puder fazer por alguém o que ele fez por mim, poderei me considerar realizado do ponto de vista profissional e humano.

Ao Prof. Nelson Maculan, por ter me recebido e encaminhado na COPPE-UFRJ e por estar sempre disposto a ajudar.

Ao Prof. Laurence Wolsey, do *Center for Operations Research and Econometrics*, da *Université Catholique de Louvain*, Bélgica, por ter viabilizado a minha visita àquela instituição, como aluno de doutorado visitante, durante o ano acadêmico de 2004/05.

Ao Prof. Andrés Weintraub, do *Departamento de Engenieria Industrial* da *Universidad de Chile*, por gentilmente ter apresentado uma aplicação interessante, inserida no contexto desta Tese.

Ao Prof. Ricardo Utsch, do Departamento de Engenharia Mecânica da UFMG, pelo incentivo desde os tempos de mestrado.

Ao Prof. Carlos Carvalho, do Departamento de Engenharia de Produção da UFMG, por ter *inoculado* o vírus da Otimização Combinatória e Inteira.

Ao Prof. Mauricio de Souza, do Departamento de Engenharia de Produção da UFMG, pelas contribuições dadas na implementação de heurísticas.

Aos meus pais, pelo amor, amizade e pelo apoio incondicional. Sem vocês, nada disso seria possível.

Ao meu irmão, pela grande torcida.

Ao meu tio e padrinho Aloísio, pelo apoio, por incentivar e por se orgulhar da minha escolha profissional.

Ao amigo Luiz Thomaz, boa prosa, para os momentos fáceis e difíceis também.

Aos funcionários, alunos e demais professores do Programa de Engenharia de Sistemas e Computação da UFRJ, por criarem um ambiente apropriado ao desen-

volvimento deste trabalho.

Finalmente, ao Conselho Nacional de Desenvolvimento Científico e Tecnológico que (através do processo 140767/2002-9) financiou a parte do meu doutorado que foi realizada no Brasil e à Capes que (através do processo BEX-1543/04-0) financiou o meu Estágio de Doutorado na *Université Catholique de Louvain*, Bélgica.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ÁRVORES ÓTIMAS EM GRAFOS: MODELOS, ALGORITMOS E APLICAÇÕES

Alexandre Salles da Cunha
Maio/2006

Orientador: Abilio Pereira de Lucena Filho
Programa: Engenharia de Sistemas e Computação

Nesta Tese, apresentamos modelos, formulações e algoritmos para problemas que, a menos de algumas restrições adicionais, podem ser entendidos como problemas de determinação de árvores ótimas em grafos. Três problemas foram estudados: o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices (PARG), o Problema da Árvore de Steiner com Recolha de Prêmios (PASRP) e, finalmente, o Problema de Planejamento da Extração de Madeira (PPEM). Apresentamos um estudo poliedral do PARG que nos permitiu introduzir algoritmos exatos e heurísticas pelo menos tão bons quanto os melhores da literatura. Após procedermos a uma reformulação do PASRP, introduzimos uma heurística Lagrangeana competitiva, com vários ingredientes novos. Introduzimos uma nova modelagem do PPEM que poderá permitir que instâncias reais, de maiores dimensões, sejam resolvidas na otimalidade. Uma contribuição adicional desta Tese foi um procedimento que propusemos para tentar caracterizar um ponto extremo de uma árvore geradora. Tal procedimento pode ser estendido a problemas que admitam relaxações definidas sobre matróides e permite transportar limites duais Lagrangeanos para um algoritmo de Planos de Corte sem a necessidade de resolver qualquer problema de separação.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

OPTIMAL TREES IN GRAPHS: MODELS, ALGORITHMS AND APPLICATIONS

Alexandre Salles da Cunha

May/2006

Advisor: Abilio Pereira de Lucena Filho
Department: Systems Engineering and Computing

In this Thesis, we presented models, formulations and algorithms for problems that can be viewed as of finding optimal trees, restricted to additional constraints. Three problems were studied: The Degree-Constrained Minimal Spanning Tree Problem, PARG, the Prize-Collecting Steiner Problem in Graphs, PASRP, and the Wood Harvesting Planning Problem, PPEM. The polyhedral study we carried out for PARG lead to algorithms at least as good as the best heuristics and exact approaches known in the literature. After reformulating the PASRP, we introduced a competitive Lagrangian heuristic for the problem. We also presented a new graph-theoretical model for PPEM which may allow the resolution of real life instances of practical sizes. A further contribution of this Thesis was the introduction of a procedure to attempt to characterize an extreme point of the Spanning Tree polytope, in order to carry Lagrangian dual bounds to a Cutting Plane algorithm without the need of any separation algorithm. Such procedure can be extended to other problems that have relaxations defined over matroids.

Sumário

1 Introdução **1**

1.1 Organização e contribuições da Tese 2

2 Otimização Inteira **6**

2.1 Introdução 6

2.2 Algoritmos de Planos de Corte 8

2.3 Relaxação Lagrangeana 12

2.4 Algoritmos *Relax and Cut* 16

2.5 Algoritmos *Branch and Bound* 19

2.6 Algoritmos *Branch-and-Cut* 22

3 Um estudo poliedral do Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices **25**

3.1 Introdução 25

3.2 Uma formulação de Programação Linear Inteira para o PARG 26

3.3 Desigualdades válidas para o PARG 27

3.3.1 Desigualdades obtidas a partir do politopo dos b-matching . . 27

3.3.2 Desigualdades generalizadas a partir do Problema do Caixeiro Viajante 29

3.3.2.1 Desigualdades Combs 30

3.3.2.2 Desigualdades Clique-Trees 32

3.4 O estudo das dimensões de desigualdades válidas para o k-PARG . . 41

3.4.1 Dimensões das faces induzidas por desigualdades da formulação clássica 42

3.4.2 Dimensão das faces induzidas pelas desigualdades Blossom . . 46

3.4.3 Dimensões das faces induzidas por desigualdades Combs e Clique Trees 54

3.5 Comentários e perspectivas 55

4 Um algoritmo Relax-and-Cut para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau 56

4.1 Introdução 57
4.2 Limites Lagrangeanos para o PARG 59
4.2.1 Separação de desigualdades Blossom em um algoritmo NDRC 60
4.2.2 Limites superiores 61
4.2.3 Pré-processamento, testes de fixação de variáveis e ciclos de MS 66
4.3 Atalhos Lagrangeanos para algoritmos de Planos de Corte 67
4.4 Experimentos computacionais 72
4.4.1 Resultados computacionais 74
4.5 Comentários finais 85

5 Um algoritmo híbrido Relax-and-Cut / Branch-and-Cut para o PARG 87

5.1 Introdução 87
5.2 O algoritmo Branch-and-Cut 88
5.2.1 Desigualdades violadas 88
5.2.1.1 A separação de desigualdades cutsets 89
5.2.1.2 Separação exata de desigualdades SEC 90
5.2.1.3 Separação heurística de desigualdades Blossom . . . 91
5.2.2 O gerenciamento do gerador de cortes e do cut-pool 93
5.2.3 Detalhes de implementação 94
5.3 Experimentos Computacionais 96
5.4 Comentários finais 103

6 Uma heurística Lagrangeana para o Problema de Steiner em Grafos com Recolha de Prêmios 106

6.1 Introdução 106
6.2 Uma formulação de Programação Inteira para o PASRP 110
6.2.1 Uma estrutura adequada para decomposição Lagrangeana . . 111
6.3 Limites Lagrangeanos para o PASRP 114
6.3.1 Limites inferiores para o PASRP 114

6.3.2 Permitindo a sobrevida de desigualdades dinamicamente dualizadas . . . 116
6.3.3 Limites superiores para o PASRP . . . 117
6.3.3.1 Uma heurística construtiva *multi-start* para o PASRP 117
6.3.3.2 O procedimeto de Busca Local . . . 119
6.3.3.3 Testes de dominância na Busca Local . . . 119
6.4 Testes de redução e de fixação de variáveis . . . 122
6.4.1 Testes de Redução . . . 122
6.4.1.1 Testes de caminho mínimo . . . 123
6.4.1.2 Teste de grau unitário . . . 123
6.4.1.3 Teste de grau dois . . . 123
6.4.1.4 Teste de grau maior-que-dois . . . 123
6.4.1.5 Testes de mínima adjacência . . . 124
6.4.1.6 Teste de ganho líquido de comprimento dois (TGLC2) 124
6.4.2 Testes de fixação de variáveis . . . 125
6.5 Experimentos computacionais com a Heurística Lagrangena . . . 126
6.5.1 Conjuntos de instâncias testados . . . 127
6.5.2 Resultados computacionais da heurística Lagrangeana . . . 128
6.6 Comparando Branch-and-Cut com Relax-and-Cut . . . 142
6.6.1 O algoritmo Branch-and-Cut . . . 142
6.6.1.1 A separação de GSECs . . . 142
6.6.1.2 Detalhes de implementação . . . 143
6.6.2 Comparação de resultados computacionais . . . 145
6.7 Comentários . . . 152

7 Contribuições para a Modelagem e Solução Exata do Problema de Planejamento de Extração de Madeira 154
7.1 Introdução . . . 154
7.2 Revisão da Literatura . . . 155
7.2.1 Abordagens baseadas em Programação Inteira . . . 155
7.2.2 Outras abordagens . . . 160
7.3 Modelos restritos às características essenciais do PPEM . . . 162
7.3.1 Formulações de Programação Inteira para os modelos propostos 165
7.3.1.1 Formulações para M1 . . . 165

7.3.1.2 Uma formulação para M2 168
7.4 Um algoritmo Branch-and-cut para a formulação GSEC de M1 169
7.4.1 Heurísticas primais 170
7.4.2 Experimentos computacionais 170
7.4.2.1 Conjuntos de problemas testados 170
7.4.2.2 Resultados computacionais preliminares 172
7.5 Comentários finais e perspectivas 174

8 Conclusões **176**

Referências Bibliográficas **179**

A Separação exata de desigualdades de eliminação de subrotas **191**

B A separação exata de desigualdades Blossom **194**

Nomenclatura e notação **198**

Lista de Figuras

3.1 Um ponto extremo $\overline{x} \in P_1(n,b), \overline{x} \notin CM(n,b,\mathbb{1})$ para o 4-PARG. . . . 30
3.2 Um ponto extremo $\overline{x} \in P_2(12,b)$ que viola uma desigualdade Comb. . 33
3.3 Representação dos conjuntos de uma Clique-Tree. 34
3.4 A operação de separação no tooth T e no handle H. 35
3.5 A operação de separação no handle. 35
3.6 Desigualdades Clique-Trees e Combs violadas por $\overline{x} \in P_2(20,b)$ 39
3.7 A desigualdade Clique-Tree ainda é violada por uma segunda solução $\overline{x} \in P_2(20,b)$. 40
3.8 Solução fracionária final $\overline{x} \in P_2(20,b)$ 40
3.9 Soluções viáveis x^1, x^2 satisfazendo $x(\delta(k+1)) = k$ 46
3.10 Soluções viáveis x^3, x^4 satisfazendo $x(\delta(k+1)) = k$ 46
3.11 Soluções viáveis x^5, x^6 satisfazendo $x(\delta(k+1)) = k$ 47
3.12 Soluções viáveis x^7, x^8, satisfazendo $x(\delta(k+1)) = k$ 47
3.13 Soluções viáveis x^9, x^{10}, satisfazendo $x(\delta(k+1)) = k$ 47
3.14 Soluções viáveis $\tilde{x}^1, \tilde{x}^2$ satisfazendo a desigualdade Blossom na igualdade 50
3.15 Soluções viáveis $\tilde{x}^3, \tilde{x}^4$ satisfazendo a desigualdade Blossom na igualdade 51
3.16 Soluções viáveis $\hat{x}^1, \hat{x}^2$ satisfazendo a desigualdade Blossom na igualdade 51
3.17 Solução viável $\tilde{x}^3$ satisfazendo a desigualdade Blossom na igualdade . 52
3.18 Solução viável $\tilde{x}^4$ satisfazendo a desigualdade Blossom na igualdade . 52
3.19 Soluções viáveis x^1, x^2, x^7 satisfazendo a desigualdade Blossom na igualdade 53
3.20 Soluções viáveis x^3, x^4 satisfazendo a desigualdade Blossom na igualdade 53
3.21 Soluções viáveis x^5, x^6 satisfazendo a desigualdade Blossom na igualdade 54

4.1 Exemplo da aplicação do algoritmo de separação de BIs. 63

6.1 Uma ASRP no grafo expandido G' 113
6.2 Uma solução $\overline{x}^k$ inviável para o PASRP 116

6.3 Exemplo onde o TGLC2 elimina a aresta (i, j) 125
6.4 Exemplo onde o TGLC2 não elimina a aresta (i, j) 125

7.1 Uma solução no digrafo D para o modelo M1 de PPEM 168

Lista de Tabelas

4.1 Parâmetros de controle empregados no algoritmo NDRC 75
4.2 Limites NDRC para as instâncias MGRAPH. 76
4.3 Comparação de pré-processamento: instâncias R, $|V| \in \{100, 500, 800\}$ 77
4.4 Resultados para instâncias R, $|V| \in \{900, 1000\}$ 78
4.5 Limites NDRC: instâncias D-E. 80
4.6 Limites avançados: instâncias D-E. 81
4.7 Limites NDRC: instâncias D-R. 82
4.8 Limites avançados: instâncias D-R. 84

5.1 Variantes testadas de algoritmos Branch-and-Cut 96
5.2 Comparação das variantes de algoritmos Branch-and-Cut: média para instâncias do grupo R 98
5.3 Comparação das variantes de algoritmos Branch-and-Cut: instâncias D-E 99
5.4 Comparação das variantes de algoritmos Branch-and-Cut: instâncias D-R 100
5.5 Resultados consolidados BC4: instâncias D-E 101
5.6 Resultados consolidados BC4: instâncias D-R 102

6.1 Consolidação de resultados de pré-processamento 128
6.2 Resultados do pré-processamento - Instâncias P e K 129
6.3 Resultados do pré-processamento - Instâncias C 130
6.4 Resultados do pré-processamento - Instâncias D 131
6.5 Resultados do pré-processamento - Instâncias E 132
6.6 Resultados consolidados da Heurística Lagrangeana 134
6.7 Resultados da Heurística Lagrangeana - Instâncias P e K 135
6.8 Resultados da Heurística Lagrangeana - Instâncias C 136
6.9 Resultados da Heurística Lagrangeana - Instâncias D 137

6.10 Resultados da Heurística Lagrangeana - Instâncias E 138
6.11 Resultados da Heurística Lagrangeana - Instâncias H 139
6.12 Sensibilidade dos limites inferiores à sobrevida das desigualdades . . . 141
6.13 Resultados do algoritmo Branch-and-Cut, Instâncias P e K 146
6.14 Resultados do algoritmo Branch-and-Cut, Instâncias C 147
6.15 Resultados do algoritmo Branch-and-Cut, Instâncias D 148
6.16 Resultados do algoritmo Branch-and-Cut, Instâncias E 149
6.17 Comparação entre a heurística Lagrangena NDRC e o algoritmo Branch-and-Cut 150
6.18 Comparação dos dois algoritmos para as instâncias resolvidas pelo algoritmo Branch-and-Cut que não foram resolvidas pela heurística Lagrangeana 152

7.1 Dimensões das instâncias 171
7.2 Resultados computacionais preliminares - Formulação de Steiner para o PPEM 174

Lista de Algoritmos

2.1 k-ésima iteração do Método do Subgradiente 15
4.1 Algoritmo de Separação de Desigualdades Blossom para NDRC . . . 62
5.1 Heurística de Separação de desigualdades Blossom, no algoritmo Branch-and-Cut para o PARG 92
B.1 Algoritmo LRT para separação exata de desigualdades Blossom [82] . 197

Capítulo 1

Introdução

Nesta Tese, estudamos problemas que, a menos de algumas restrições adicionais, podem ser entendidos como problemas de determinação de árvores ótimas em grafos. Problemas desta natureza estão entre os mais estudados em Otimização Combinatória. Isto se deve à beleza do assunto em si mesmo, à estreita relação destes com vários outros problemas na área e ao fato de que árvores são objetos matemáticos com enorme capacidade de modelar uma grande variedade de problemas reais cuja solução desperta grande interesse econômico.

Em particular, estamos interessados no estudo do Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices, identificado aqui por PARG; do Problema de Steiner em Grafos com Recolha de Prêmios, identificado por PASRP; e, finalmente, do Problema de Planejamento de Extração de Madeira, denominado de PPEM.

No PARG, pretende-se encontrar uma Árvore Geradora de Custo Mínimo na qual o *número de arestas incidentes* em cada vértice seja inferior ao *limite de grau* previamente estabelecido para o vértice. Aplicações deste problema podem ser encontradas, por exemplo, no projeto de redes de computadores [47], de circuitos elétricos [102], dentre outras.

Dado um grafo não direcionado no qual são associados *custos* às arestas e *penalidades* aos vértices, no PASRP, o *peso* de uma árvore equivale à soma dos custos de suas arestas mais a soma das penalidades dos vértices por ela não cobertos. Uma solução ótima para o problema é uma árvore, não necessariamente geradora, de mínimo peso. Aplicações do PASRP podem ser encontradas, principalmente, quando há uma relação clara de compromisso entre realizar os investimentos necessários para incluir um vértice adicional a uma solução ou incorrer em uma penalidade por

deixá-lo fora dela. Dentre as aplicações mais conhecidas, podemos citar o planejamento de distribuição de gás natural e energia elétrica [85] e o projeto de redes de fibra ótica para prover serviços de conexão de banda larga a clientes comerciais e residenciais [92].

Para melhor compreender o PPEM [121], considere que uma região florestal é dividida em *células* que representam áreas a serem colhidas, isto é, áreas que possuem quantidade significativa de árvores. Adicionalmente, há, na floresta, alguns pontos onde podem ser instaladas máquinas que serão responsáveis por efetuar a colheita das células. O maquinário é necessário para transportar as toras de madeira dos locais onde foram derrubadas até alguns outros pontos na floresta, que se conectam a uma rede de estradas. Esta por sua vez, contendo trechos existentes ou a serem construídos, deve permitir que a madeira colhida alcance um ponto de saída da floresta. A partir de então a madeira será levada aos pontos de consumo ou de beneficiamento. No PPEM, deseja-se resolver o problema combinado de localização de maquinário e projeto da rede de estradas, de modo que o lucro associado à colheita seja máximo. Naturalmente, o PPEM assume especial importância em países onde a Indústria Florestal responde por parcela significativa do produto nacional, como por exemplo, Chile, Canadá, Suécia, Finlândia e Nova Zelândia [40, 116].

1.1 Organização e contribuições da Tese

Tratamos na Tese de problemas distintos que envolvem uma estrutura comum. Para resolvê-los, empregamos Relaxação Lagrangeana [42, 62, 49] e algoritmos poliedrais do tipo Branch-and-Cut [108], em conjunto ou separadamente. Assim sendo, no Capítulo 2, apresentamos uma sucinta revisão sobre os principais aspectos teóricos e algoritmicos associados à estas técnicas de solução. Também no Capítulo 2, introduzimos a notação empregada ao longo do texto.

Nos três Capítulos seguintes da Tese, nos concentramos no estudo do PARG. Embora tenha tenha sido amplamente estudado na literatura [126, 44, 102, 47, 110, 117, 127, 122, 73, 25, 74, 76, 15, 114, 1], nenhum trabalho anterior a esta Tese tratou do PARG do ponto de vista poliedral. No Capítulo 3, apresentamos uma contribuição neste sentido. Apresentamos inicialmente uma formulação de Programação Inteira clássica para o problema que foi a base de todos os trabalhos divulgados anteriormente. Em seguida, mostramos que desigualdades válidas para

politopos correlatos, como por exemplo, as desigualdades Blossom [35], Combs [58] e Clique-Trees [60], possuem contrapartidas que são válidas para o PARG. Provamos que estas desigualdades fortalecem a formulação clássica do problema. Encerrando o Capítulo, apresentamos um estudo das dimensões das faces que estas desigualdades induzem no politopo associado ao k-PARG, um caso particular do PARG.

No Capítulo 4, pela primeira vez em 25 anos de história do problema, fomos capazes de fortalecer, pela dualização de desigualdades Blossom em um algoritmo Lagrangeano do tipo Relax-and-Cut [89], os limites inferiores dados pela formulação clássica do PARG. Os resultados computacionais obtidos indicam que os limites inferiores aqui propostos dominam ou são competitivos com os limites Lagrangeanos ou de Relaxação Linear encontrados na literatura. Além disto, nossos limites superiores são competitivos com os melhores da literatura.

Numa segunda contribuição do Capítulo 4, propusemos um procedimento para tentar caracterizar os hiperplanos cuja interseção define um dado ponto extremo do politopo das árvores geradoras. Este procedimento permite transportar nossos limites duais Lagrangeanos diretamente para um algoritmo de Planos de Corte. Isto é feito de forma extremamente conveniente, não exigindo a resolução de qualquer problema de separação. Apesar de ter sido proposto no contexto de árvores geradoras, este procedimento pode ser estendido, de forma direta, para problemas que admitem relaxações definidas sobre matróides.

No Capítulo 5 apresentamos um algoritmo híbrido Relax-and-Cut / Branch-and-Cut para a resolução exata do PARG. O algoritmo é composto por duas etapas: uma fase de pré-processamento (baseada no algoritmo Relax-and-Cut apresentado no Capítulo 4) e um algoritmo Branch-and-Cut no qual várias classes de desigualdades válidas para o PARG são separadas. Em uma avaliação preliminar, poder-se-ia dizer que o algoritmo proposto aqui assemelha-se ao de Caccetta e Hill [15], o único algoritmo Branch-and-Cut então conhecido na literatura para o PARG. Entretanto, como veremos, em suas duas fases, nosso algoritmo contém ingredientes que o diferenciam bastante do algoritmo em [15]. Em linhas gerais, o nosso algoritmo se diferencia por empregarmos desigualdades Blossom tanto na fase Relax-and-Cut quanto na fase de Planos de Corte. Se diferencia também por utilizarmos uma heurística primal que contém uma busca local, ao longo de toda a árvore de enumeração. No entanto, provavelmente, a diferença mais marcante entre os dois algoritmos reside no fato de utilizarmos limites inferiores gerados pelo algoritmo Relax-and-Cut para

uma inicialização avançada, sem necessidade de resolver problemas de separação, de um algoritmo de Planos de Corte. Os resultados computacionais que obtivemos indicam que o uso das desigualdades Blossom e do procedimento que permitiu a inicialização avançada do algoritmo de Planos de Corte contribuiram fortemente para a melhoria de desempenho do método proposto. Assim sendo, acreditamos que nosso algoritmo exato domina aquele em [15] que não incorpora estas idéias.

No Capítulo 6, introduzimos uma heurística Lagrangeana para o PASRP. No esquema aqui proposto, o método de Relaxação Lagrangeana é aplicado à formulação do PASRP introduzida em [92]. À primeira vista, esta formulação não parece adequada à utilização de Relaxação Lagrangeana. Procedemos então a uma reformulação da mesma, introduzindo um novo conjunto de variáveis que confere ao PASRP uma interpretação em grafos mais conveniente. Sob essa reformulação, o PASRP torna-se bastante atraente para ser decomposto de forma Lagrangena e um algoritmo Relax-and-Cut é assim descrito para o problema. Este algoritmo emprega informação dual Lagrangeana para direcionar a geração de soluções viáveis para o PASRP. Essa heurística, que envolve uma fase construtiva e uma Busca Local, é chamada repetidas vezes ao longo da execução do algoritmo Relax-and-Cut. A Busca Local emprega testes de dominância para o PASRP que foram adaptados de testes propostos na literatura do Problema de Steiner em Grafos. O algoritmo Relax-and-Cut, por sua vez, emprega também testes de pré-processamento e de fixação de variáveis. De particular interesse, é um novo teste de pré-processamento que introduzimos para o problema. Concluindo o Capítulo, implementamos um algoritmo Branch-and-Cut baseado na formulação do PASRP introduzida em [92] e o comparamos com a heurística Lagrangeana aqui sugerida. Os resultados computacionais obtidos indicam que a heurística introduzida produziu resultados compatíveis com os esperados de um algoritmo desta classe, isto é, obteve uma boa relação de compromisso entre a qualidade dos limites encontrados e o tempo de CPU empregado.

No Capítulo 7, apresentamos contribuições para o modelamento e solução exata do PPEM. Após adotarmos algumas hipóteses que simplificam a topologia das soluções associadas ao problema, introduzimos dois modelos com representações em grafos para o mesmo. No primeiro, o PPEM pode ser entendido como um Problema de Steiner em Grafos, sujeito à restrições adicionais. No segundo, uma variante do Problema de Steiner em Grafos com Recolha de Prêmios sujeito à restrições adicionais é proposto. Para cada um desses modelos, formulações de Programação Inteira

são introduzidas. Implementamos um algoritmo Branch-and-Cut para uma delas. Embora limitados, os resultados computacionais preliminares que obtivemos sugerem que o refinamento algoritmico da abordagem proposta poderá contribuir para a solução de instâncias de interesse prático.

Finalmente, concluimos a Tese no Capítulo 8, revisando os principais resultados obtidos.

Ao longo da Tese, várias abreviaturas serão utilizadas para nos referirmos aos problemas aqui tratados e a outros a eles relacionados. Assim sendo, ao final da Tese, apresentamos uma lista das principais abreviaturas, bem como dos símbolos matemáticos empregados ao longo do texto.

Capítulo 2

Otimização Inteira

2.1 Introdução

Neste Capítulo, serão descritos os métodos e técnicas de otimização inteira utilizados na Tese. Sem perda de generalidade, fazemos a apresentação assumindo que os problemas a serem aqui resolvidos encontram-se na forma de minimização. Inicialmente, apresentamos alguns conceitos elementares que serão usados ao longo do texto. Posteriormente, abordaremos especificamente os métodos e técnicas que iremos utilizar nos capítulos subsequentes.

Um dos primeiros passos para resolver um Problema de Otimização Linear Inteiro (PI) consiste em obter uma *formulação matemática* para o mesmo. Se o problema for $\mathcal{NP}$-difícil [46], é importante não dissociar essa formulação da técnica de solução a ser empregada para resolvê-la. Os dois devem formar um par harmônico onde a técnica de solução deve ser capaz de melhor explorar as especificidades da formulação.

Definição 2.1 *Dado um PI*

$$z = min\{c^t x : Ax \leq b, x \in \mathbb{Z}_+^n\}, \tag{2.1}$$

onde $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m, c \in \mathbb{R}^n$, *dizemos que* $P = \{Ax \leq b, x \in \mathbb{Z}_+^n\}$ *define uma formulação para o problema, se um vetor de incidência* $x_i \in \mathbb{Z}_+^n$ *associado a qualquer solução viável de PI satisfaz* $Ax_i \leq b$ *e se, adicionalmente, não existe um vetor de incidência* $y \in \mathbb{Z}_+^n$ *não viável para PI, tal que* $Ay \leq b$.

Apresentamos a seguir uma estrutura bastante importante para mensurar a qualidade de uma formulação para um PI.

Definição 2.2 *Dado PI, a envoltória convexa (conv(PI)) dos p vetores de incidência* $\{x_i : i = 1, ..., p\}$ *associados às suas soluções é:*

$$conv(PI) = \{x \in \mathbb{R}^n : x = \sum_{i=1}^{p} \mu_i x_i, \quad \sum_{i=1}^{p} \mu_i = 1, \quad \mu_i \in \mathbb{R}_+, \; i = 1, \ldots, p\}.$$

Em função da natureza discreta das variáveis envolvidas, para se resolver um PI, é necessário apresentar um limite superior (*primal*) $\overline{z}$, dado pelo custo de uma solução viável conhecida para o problema, e um limite inferior (*dual*) $\underline{z}$, associado a uma *relaxação válida* para o mesmo, de forma que $\overline{z} = \underline{z}$.

Definição 2.3 *(Geoffrion [49]) Dado o PI (2.1), dizemos que*

$$\underline{z} = min\{d^t x : x \in F\} \tag{2.2}$$

onde $d \in \mathbb{R}^n$ *e* $F \subseteq \mathbb{R}^n_+$*, é uma relaxação válida para PI se e somente se:*

1. $P \subseteq F$
2. $d^t x \leq c^t x, \forall x \in P.$

Neste caso, dizemos que $\underline{z}$ *é um limite inferior válido para* z *e chamamos (2.2), genericamente, de uma Relaxação (R) de PI.*

Para a obtenção de soluções viáveis para um PI, pode-se recorrer a heurísticas, ou seja, procedimentos que são idealmente capazes de encontrar uma solução viável (não necessariamente ótima) para o problema tratado, em um tempo de CPU *aceitável*. As heurísticas propostas nesta Tese são específicas para cada problema e serão portanto discutidas nos Capítulos referentes a cada um destes.

Limites duais são obtidos através de algoritmos que resolvem uma dada relaxação R de PI. Note, entretanto, que assim procedendo não temos uma garantia de que ao final da execução do algoritmo uma solução viável será encontrada para PI. Tais algoritmos fornecem entretanto, limites inferiores válidos para o problema que se tenta resolver.

Nas próximas Seções deste Capítulo discutimos a geração de limites duais para PI. Entretanto, antes de prosseguir, cabe destacar que o objetivo de nossa apresentação não é esgotar o assunto. Pretendemos apenas apresentar uma breve introdução ao mesmo, explicitando os principais aspectos teóricos e algorítmicos envolvidos. Ao longo desse processo, iremos também introduzir a notação utilizada na Tese.

Este Capítulo é organizado da seguinte forma. Suas três primeiras Seções concentram a apresentação de técnicas algorítmicas para a obtenção de limites inferiores válidos. São eles: Algoritmos de Planos de Corte na Seção 2.2, Algoritmos Baseados em Relaxação Lagrangeana Clássica na Seção 2.3 e Algortimos Relax-and-Cut na Seção 2.4. Já nas duas Seções seguintes, discutimos como os mecanismos de obtenção de limites duais são empregados em algoritmos exatos do tipo Branch-and-Bound, na Seção 2.5, e Branch-and-Cut, na Seção 2.6.

2.2 Algoritmos de Planos de Corte

Considere o seguinte Problema de Programação Linear

$$z_{PL} = \min\{c^t x : x \in S\}, \tag{2.3}$$

onde $S = \{Ax \leq b, x \in \mathbb{R}^n_+\}$. Uma vez que as condições de integralidade anteriormente impostas a x foram agora substituídas por $\{x \in \mathbb{R}^n_+\}$, o problema acima fornece uma Relaxação válida (dita *Relaxação de Programação Linear*, abreviada PL) para PI. Seja então $\overline{x} \in \mathbb{R}^n_+$ uma solução de PL. Se $\overline{x}$ é inteira, temos que $c^t\overline{x}$ é simultaneamente um limite superior e um limite inferior válidos para PI e, portanto, $\overline{x}$ resolve o problema.

Observação 2.1 *Uma formulação P para PI é tanto melhor quanto mais próxima sua Relaxação Linear for (no sentido geométrico) de $conv(PI)$.*

Note que se empregarmos uma Relaxação Linear definida em uma região de viabilidade S que coincida com $conv(PI)$ e se utilizarmos o Método Simplex para resolvê-la, temos a garantia de que $\overline{x}$ resolve PI, já que, por definição, todos os vértices de $conv(PI)$ são inteiros. Para os casos gerais de Problemas de Otimização Combinatória e Inteira, obter relaxações satisfazendo a esta desejável propriedade é bastante improvável [72]. Entretanto, podemos empregar técnicas que aprimorem a *aproximação* de $conv(PI)$, dada pelo poliedro S. Vale também ressaltar que, em geral, para provar a otimalidade, necessitamos apenas que S e $conv(PI)$ coincidam na vizinhança de um ponto ótimo.

A idéia central dos algoritmos de Planos de Corte é exatamente esta. Para melhor entendê-la considere a Definição a seguir.

Definição 2.4 *(Desigualdade válida) Uma desigualdade* $D = \{x \in \mathbb{R}^n_+ : \pi^t x \leq \pi_0\}$ *é válida para PI, se* $\pi x_i \leq \pi_0$ *para qualquer solução* x_i *viável para PI. De forma análoga, uma classe* $\mathcal{D}$ *de desigualdades é válida para PI se cada desigualdade na classe for válida para PI.*

Suponha agora que $\overline{x}$, a solução de uma Relaxação Linear de PI, não o resolva. Nos algoritmos de Planos de Cortes, procura-se identificar uma desigualdade válida para PI, por exemplo $D = \{x \in \mathbb{R}^n_+ : \pi^t x \leq \pi_0\}$, que *corte* $\overline{x}$, isto é que satisfaça $\pi^t \overline{x} > \pi_0$. A esta restrição linear damos o nome de *plano de corte* ou, simplesmente, *corte*. Uma vez identificados, estes cortes são introduzidos em PL, na forma de restrições. Uma nova região S, agora mais restrita, é então obtida.

Note que quando incorporamos os cortes, obtendo uma nova região poliedral S, preservamos todos os vetores de incidência associados às soluções viáveis do problema, mas *cortamos* o ponto $\overline{x}$, no sentido de que este ponto não mais satisfaz a descrição poliédrica corrente do problema. Na verdade, esta estratégia sistematicamente aproxima, pelo menos na vizinhança de $\overline{x}$, a descrição poliedral de $conv(PI)$. Assim sendo, uma nova relaxação válida PL e um novo limite inferior z_{PL} podem ser obtidos com a introdução de cortes.

Observe ainda que, uma vez que a região de viabilidade da nova relaxação está contida na região de viabilidade da relaxação anterior, é certo que o novo limite inferior seja pelo menos tão bom quanto aquele que o precedeu. Idealmente, deseja-se que estes limites aumentem com a introdução de cortes.

Provavelmente, os primeiros algoritmos de Planos de Corte que se tem notícia são devidos a Dantzig et al. [30] e Gomory [55].

Informalmente, a idéia inicial de Gomory [55] (*cortes de Gomory*) era gerar um corte a partir do arredondamento inteiro obtido de uma linha (no Tableau do Método Simplex) associada a uma variável básica fracionária da solução $\overline{x}$ da Relaxação Linear PL de PI. Note que, dado um vetor $\lambda \in \mathbb{R}^m_+$, a desigualdade $\{x \in \mathbb{R}^n_+ : \lfloor \lambda A x \rfloor \leq \lfloor \lambda b \rfloor\}$ é válida para PI. Formalmente, no método de Planos de Cortes de Gomory, deseja-se obter um vetor de coeficientes λ tal que $\lfloor \lambda A \overline{x} \rfloor > \lfloor \lambda b \rfloor$. Estes cortes são denominados Cortes de Gomory.

Teoricamente, os algoritmos baseados em Planos de Corte de Gomory convergem (veja [55]), isto é, após a introdução de um número finito de cortes, a solução da correspondente Relaxação Linear resolve o problema original PI. Um exemplo bem sucedido da aplicação da técnica de Planos de Corte para se resolver proble-

mas de Otimização Combinatória foi descrita por Miliotis [97], ao tratar o Problema do Caixeiro Viajante Simétrico (PCV). Naquele trabalho, como já observado anteriormente, o autor salientou que, na prática, os algoritmos baseados em Planos de Corte de Gomory podem apresentar problemas numéricos decorrentes do crescimento exagerado dos coeficientes dos cortes, ao longo da aplicação. Por este motivo, por algum tempo, existiu uma certa descrença da comunidade científica em relação à praticidade da utilização da técnica de Planos de Corte, como um instrumento efetivo para resolver Problemas de Otimização Inteira.

Este quadro começou a mudar quando novas técnicas para identificação de desigualdades válidas passaram a ser empregadas sistematicamente. Informalmente, este processo já havia sido iniciado com o algoritmo de Planos de Corte de Dantzig et al. [30]. Estas técnicas, fundamentadas no estudo das *desigualdades lineares necessárias* para caracterizar $conv(PI)$, inaugurou uma nova área de estudo em Otimização Inteira e Combinatória, denominada *Combinatória Poliédrica.*

O objetivo primordial destes estudos consiste em caracterizar analiticamente classes de desigualdades válidas *fortes* para PI. Para facilitar a compreensão deste conceito, considere as definições seguintes.

Definição 2.5 *(Equivalência) Sejam* $D_1 = \{x \in \mathbb{R}^n_+ : \pi^t x \leq \pi_0\}$ *e* $D_2 = \{x \in \mathbb{R}^n_+ : \gamma^t x \leq \gamma_0\}$ *duas desigualdades onde* $\pi, \gamma \in \mathbb{R}^n$ *e* $\pi_0, \gamma_0 \in \mathbb{R}$. *Dizemos que* D_1 *e* D_2 *são equivalentes, se existe* $u > 0$ *tal que* $\pi = u\gamma$ *e* $\pi_0 = u\gamma_0$.

Observe que, geometricamente, duas desigualdades são equivalentes se induzem o mesmo subespaço de $\mathbb{R}^n$.

Definição 2.6 *(Dominância) Considere* D_1 *e* D_2 *definidas anteriormente.* D_1 *domina (ou é mais forte que)* D_2 *se:*

1. D_1 *e* D_2 *não são equivalentes;*

2. *existe* $u > 0$ *tal que* $\pi \geq u\gamma$ *e* $\pi_0 \leq u\gamma_0$.

Recorrendo novamente a argumentos geométricos, observe que, se D_1 domina D_2, temos que $\{x \in \mathbb{R}^n_+ : \pi^t x \leq \pi_0\} \subset \{x \in \mathbb{R}^n_+ : \gamma^t x \leq \gamma_0\}$.

As próximas definições nos auxiliam a estabelecer a relação de dominância e equivalência de desigualdades em relação a um poliedro.

Definição 2.7 *(Face) Dizemos que S_D define (ou induz) uma face de um poliedro S, se $S_D = \{x \in S : \pi x = \pi_0\}$ para alguma desigualdade $D = \{x \in \mathbb{R}^n_+ : \pi x \leq \pi_0\}$ válida para S.*

Definição 2.8 *(Face própria) Assuma que $S_D = \{x \in S : \pi x = \pi_0\}$ define uma face de S. Se $S_D \neq \emptyset$ e se $S_D \neq S$, S_D é uma face própria de S.*

Definição 2.9 *(Equivalência de desigualdades em relação a um poliedro) Considere duas desigualdades D_1 e D_2, que induzem faces próprias S_{D_1} e S_{D_2} de S, respectivamente. Se $S_{D_1} = S_{D_2}$, D_1 e D_2 representam a mesma face de S e são ditas equivalentes em relação a S.*

Observe que se D_1 e D_2 são equivalentes (ou representam a mesma face própria) em relação a (de) S, então D_1 (resp. D_2) pode substituir D_2 (resp. D_1) na caracterização linear do mesmo, sem que a região por ele definida seja alterada. Por outro lado, se D_1 e D_2 induzem faces próprias de S e $S_{D_2} \subset S_{D_1}$, D_1 torna D_2 uma desigualdade desnecessária na caracterização de S. Este conceito fica formalizado a partir das definições seguintes.

Definição 2.10 *(Dimensão de um poliedro) Seja $S = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ um poliedro. A dimensão de S ($dim(S)$) é igual a uma unidade a menos que máximo número d de vetores de incidência $\{x_i \in \mathbb{R}^n_+ : i = 1, \ldots, d\}$, afim independentes, que satisfazem o sistema $Ax_i \leq b$.*

Definição 2.11 *(Faceta) Uma desigualdade $D = \{x \in \mathbb{R}^n_+ : \pi^t x \leq \pi_0\}$, válida para o poliedro S, induz uma faceta S_D de S, se e somente se S_D induz uma face própria de S e se $dim(S_D) = dim(S) - 1$.*

À luz dos conceitos apresentados até aqui, podemos dizer que o objetivo primorial em Combinatória Poliédrica consiste em caracterizar famílias de desigualdades que induzam facetas de $conv(PI)$. Estas, por sua vez, são as únicas desigualdades necessárias para descrever $conv(PI)$. Qualquer outra desigualdade válida para $conv(PI)$ que induza uma face de dimensão inferior a $dim(conv(PI)) - 1$ é dominada por uma combinação linear de desigualdades que induza uma faceta de $conv(PI)$.

Edmonds [35, 36, 37] com suas descrições lineares de importantes problemas, polinomiais, de Otimização Combinatória, pode ser considerado como o principal responsável pela disseminação da Combinatória Poliédrica. Entretanto, em grande

parte, o rápido desenvolvimento desta área, bem como sua aplicação bem sucedida em vários problemas de Otimização Inteira, resultam do trabalho pioneiro de Grötschel e Padberg [58, 59], Padberg e Grötschel [105], Crowder e Padberg [27] e Padberg e Hong [106] no estudo do poliedro do PCV.

Para identificar uma desigualdade válida D pertencente a uma classe $\mathcal{D}$, que corte uma região de viabilidade dada por uma Relaxação Linear de PI, é necessário que um *problema de separação* seja resolvido. Dizemos que um *algoritmo de separação* para $\mathcal{D}$ é exato, se tal algoritmo é garantidamente capaz de encontrar, caso exista, uma desigualdade violada (em $\mathcal{D}$) pela solução da Relaxação Linear em mãos. Caso contrário, é denominado uma heurística de separação.

Para que o uso de uma classe de desigualdades válidas seja efetivo em um algoritmo de Planos de Corte é necessário dispor de algoritmos de separação eficientes (isto é, cujo tempo de execução seja polinomial) e, preferencialmente, rápidos.

Uma aplicação importante dos algoritmos de Planos de Corte, que diz respeito aos problemas tratados nesta Tese, é a seguinte. Assuma que uma formulação contendo uma classe com exponencialmente muitas restrições seja dada para um Problema de Otimização Combinatória. Naturalmente, à medida que cresce a dimensão das instâncias do problema, torna-se inviável trabalhar com Relaxações Lineares que explicitamente incluam todas essas restrições. No entanto, diante de um algoritmo de separação adequado (para a classe de restrições mencionada acima) é possível conceber um algoritmo de Planos de Corte que nos leve a uma Relaxação Linear daquela formulação.

A seguir, apresentamos outra técnica bastante utilizada na solução de Problemas de Otimização Inteira.

2.3 Relaxação Lagrangeana

Em linhas gerais, a Relaxação Lagrangeana é um método de decomposição que nos permite explorar estruturas especiais encontradas na formulação do problema que se quer resolver. Este processo de decomposição consiste em identificar um conjunto de restrições consideradas *difíceis*, adicioná-las (ou dualizá-las) à função objetivo através de *Multiplicadores de Lagrange* e resolver o problema remanescente, agora menos restrito, definido sobre uma estrutura atraente. Em algumas formulações, estruturas atraentes são aparentes à primeira vista. Em outros casos, para que sejam

identificadas, faz-se necessário a reformulação do problema. Nesta Seção, tratamos da abordagem Clássica de Relaxação Lagrangeana. Por *abordagem clássica*, entendemos uma aplicação onde o número de desigualdades candidatas à dualização é uma função polinomial da dimensão do problema a ser resolvido e onde a dualização é realizada de forma estática, uma única vez.

Formalmente, considere que se deseja resolver o problema PI, agora definido por

$$z = \min\{c^t x : Ax \leq b, Cx \leq f, x \in \mathbb{Z}_+^n\}, \tag{2.4}$$

com $A \in \mathbb{Q}^{m \times n}, C \in \mathbb{Q}^{m_2 \times n}, b \in \mathbb{Q}^m, f \in \mathbb{Q}_2^m, c \in \mathbb{R}^n$.

Assuma que PI, na ausência das restrições $\{Ax \leq b\}$ seja de fácil resolução, no sentido de que exista um algoritmo polinomial (eficiente) capaz de resolvê-lo (ou que este problema seja *pseudo-polinomial*, como é o caso do Problema da Mochila $0-1$ [29, 11], ou até mesmo que seja $\mathcal{NP}$-difícil, mas de muito pequena dimensão). Seja $a_i \in \mathbb{R}^n$ um vetor linha da matriz A. Se associarmos à cada uma das desigualdades $\{a_i x \leq b_i : i = 1, ..., m\}$, um multiplicador de Lagrange $\lambda_i \in \mathbb{R}_+$ e levarmos o produto resultante à função objetivo, obtemos o *Problema Lagrangeano*, PL(λ), definido por:

$$z_\lambda = \min \left\{ c^t x + \lambda^t (Ax - b) : Cx \leq f, x \in \mathbb{Z}_+^n \right\}. \tag{2.5}$$

Note que PL(λ) fornece uma relaxação válida para PI. Isso pode ser constatado ao observar que:

1. a região de viabilidade do problema relaxado PL(λ) contém a região de viabilidade do problema original PI;

2. para qualquer solução viável x_i de PI e $\lambda \in \mathbb{R}_+^m$, temos $\lambda^t(Ax_i - b) \leq 0$. Portanto $c^t x_i + \lambda^t(Ax_i - b) \leq c^t x_i$.

O mesmo procedimento para obter relaxações válidas pode ser empregado caso PI seja definido em termos de um sistema de igualdades $Ax = b$, ao invés do sistema de desigualdades inicialmente apresentado. Nesse caso, podemos, equivalentemente, dualizar $Ax \leq b$ e $-Ax \leq -b$, associando multiplicadores $\lambda_1 \in \mathbb{R}_+^m$ e $\lambda_2 \in \mathbb{R}_+^m$, respectivamente, a cada um desses conjuntos de restrições. A função objetivo do Problema Lagrangeano assim obtida, ou seja, $c^t x + \lambda_1^t(Ax - b) + \lambda_2^t(-Ax + b)$, pode ser reescrita como $c^t x + (\lambda_1 - \lambda_2)^t(Ax - b)$. Note que, nesse caso, o vetor $\lambda = \lambda_1 - \lambda_2$, associado ao sistema de igualdades, será irrestrito em sinal.

Uma vez que z_λ nos fornece um limite inferior válido para z, interessa-nos conhecer um vetor $\lambda \in \mathbb{R}^m_+$ que determine o melhor limite z_λ possível. Em outras palavras, interessa-nos resolver o *Problema Dual Lagrangeano* (PDL)

$$z_d = \max_{\lambda \in \mathbb{R}^m_+} \{z_\lambda\}. \tag{2.6}$$

Um aspecto importante ao se optar por um esquema de Relaxação Lagrangeana, é saber como o limite z_d se compara ao limite z_{PL}. Os resultados a seguir (veja Geoffrion [49]) respondem à esta questão.

Teorema 2.1 *(Geoffrion [49]) O valor de z_d é igual ao valor do Programa Linear*

$$min\{c^t x : Ax \leq b, x \in conv(\{x \in \mathbb{Z}^n_+ : Cx \leq f\})\}. \tag{2.7}$$

Definição 2.12 *Se $conv(\{x \in \mathbb{Z}^n_+ : Cx \leq f\}) = \{x \in \mathbb{R}^n_+ : Cx \leq f\}$, dizemos que PDL satisfaz à Propriedade de Integralidade.*

Observação 2.2 *Note que, se PDL satisfaz à Propriedade de Integralidade, os pontos extremos do poliedro $\{x \in \mathbb{R}^n_+ : Cx \leq f\}$ são inteiros. Assim sendo, o limite z_d iguala z_{PL}. Uma conseqüência adicional do Teorema acima é que z_d é pelo menos tão bom quanto z_{PL}.*

À luz destes resultados, caso PDL não satisfaça à Propriedade de Integralidade, podemos eventualmente obter $z_d > z_{PL}$. Assim sendo, a escolha do conjunto de desigualdades a dualizar em Relaxação Lagrangeana deve levar em conta, simultaneamente, a dificuldade de resolver os Problemas Lagrangeanos associados, bem como a qualidade dos limites inferiores que podem ser obtidos. Em outras palavras, Relaxação Lagrangeana pode também ser empregada mesmo quando não se conhece um algoritmo eficiente para resolver PL(λ), mas, ainda assim, na prática, este possa ser resolvido.

Vamos agora discutir métodos para a solução do PDL. A função objetivo deste problema é contínua, estritamente quase-côncava e diferenciável por partes. Para resolvê-lo, pode-se empregar o *Método do Subgradiente* [68], o *Método de Bundle* [80], ou o *Algoritmo do Volume* [6, 3], dentre outros. Em função de sua simplicidade de implementação, o Método do Subgradiente (MS) foi empregado nos algoritmos baseados em Relaxação Lagrangeana desenvolvidos nesta Tese. O MS pode ser

entendido como uma adaptação do Método do Gradiente a problemas não diferenciáveis. Portanto, trata-se de um procedimento iterativo, no qual a cada iteração k, dado um vetor de multiplicadores λ^k, um passo é dado na direção de um subgradiente da função objetivo do Problema Lagrangeano. A iteração típica deste método é apresentada na forma do Algortimo 2.1, a seguir.

Algoritmo 2.1 k-ésima iteração do Método do Subgradiente

1: **Determine** $\overline{x}^k$, a solução de PL(λ^k)
2: **Calcule** um vetor de subgradientes $s^k \in \mathbb{R}^m$, tal que $s_i^k = a_i\overline{x}^k - b_i, i = 1, \ldots, m$
3: **Atualize** $\{\lambda_i^{k+1} = \max\{0; \lambda_i^k + t^k s_i^k\}, i = 1, ..., m\}$, onde o tamanho de passo $t^k \in \mathbb{R}_+$ é dado por

$$t^k = \rho^k \left(\frac{\overline{z} - z_{\lambda^k}}{\sum_{i=1}^m (s_i^k)^2} \right), 0 < \rho^k \leq 2 \tag{2.8}$$

4: **Faça** $k \leftarrow k + 1$

A convergência teórica da sequência $\{\lambda^k\}$ definida pelo Método do Subgradiente para o vetor de multiplicadores que resolve PDL foi amplamente estudada na literatura (veja Minoux [98], p. $120 - 123$, por exemplo).

Talvez o primeiro trabalho que tenha tratado da validação prática do Método do Subgradiente em Otimização inteira foi Held et al. [69]. De um modo geral, a convergência prática do método depende de uma boa estimativa $\overline{z}$ para o valor de z e da redução de ρ^k ao longo das iterações do método. Desta forma, é usual em uma implementação do MS, inicializar ρ^k com 2 e ir reduzindo este parâmetro à metade, sempre que o melhor limite inferior não crescer por ξ iterações consecutivas. O parâmetro ξ é usualmente conhecido como *freqüência de redução*.

Outro aspecto crucial em Relaxação Lagrangeana diz respeito à otimalidade de $\overline{x}^*$. Seja λ^* o vetor de multiplicadores que resolve PDL e $\overline{x}^*$ a correspondente solução de PL(λ^*). A solução $\overline{x}^*$ é ótima para PI se e somente se $\sum_{i=1}^m \lambda_i^*(a_i\overline{x}^* - b_i) = 0$. Não raro, esta condição não é satisfeita por $\overline{x}^*, \lambda^*$. Ainda que os valores ótimos teóricos $\overline{x}^*, \lambda^*$ resolvam PI, o Método do Subgradiente pode apresentar dificuldades em gerar uma seqüência de multiplicadores que convirja para aquele vetor ótimo. Um dos motivos que pode colaborar para este comportamento indesejável do método está associado à regra usada para a redução do passo, diante de valores muito pequenos, na medida em que a diferença $(\overline{z} - z_\lambda)$ diminui. Uma alternativa muito empregada na literatura e que, via de regra funciona bem, consiste em substituir a fórmula de cálculo de t^k indicada no Algoritmo 2.1 por $t^k = (1 + \varphi)\rho^k \left(\frac{\overline{z} - z_{\lambda^k}}{\sum_{i=1}^m (s_i^k)^2} \right)$. O uso

do parâmetro de *sobreposição* φ (normalmente assumindo valores entre 0.01 e 0.05) auxilia para que, na prática, o Método do Subgradiente não sofra uma estagnação prematura.

Estes aspectos, bem como a não diferenciabilidade de PDL em todo o seu domínio e o fato do subgradiente não ser necessariamente uma *direção de subida* dificultam a adoção de critérios de convergência similares aos empregados em outros métodos de Programação Matemática [98]. Assim sendo, uma regra de parada muito empregada para o MS consiste em limitar o número de iterações realizadas a um número máximo, denominado aqui de SMMAX.

Diversos trabalhos trataram de estratégias que podem, eventualmente, melhorar o desempenho prático do MS. Dentre aqueles que sugeriram alternativas para o cálculo do tamanho do passo, podemos citar Bazaraa [8] e Bazarra e Sherali [7]. Outros trabalhos estudaram modificações na direção de busca do método, visando dotá-lo de maior estabilidade. Nesta linha, podemos mencionar Camerini et al. [17] e Fumero [43]. Não há, porém, indicação clara de que estas regras se comportem consistentemente bem. Além disto, quase sempre, para uma instância do problema que se quer resolver, o ajuste destas regras costuma ser bastante específico.

Maiores detalhes sobre Relaxação Lagrangena e seu uso em Otimização Combinatória e Inteira podem ser obtidas em referências clássicas sobre o assunto, dentre as quais, podemos citar: Geoffrion [49], Shapiro [119], Fisher [42] e, mais recentemente, Lemarechal [81] e Guignard [62].

2.4 Algoritmos *Relax and Cut*

Como mencionado anteriormente, em Relaxação Lagrangeana Clássica, o número de desigualdades candidatas à dualização Lagrangeana é uma função polinomial das dimensões de entrada do problema. Em termos da notação da Seção anterior, m é uma função polinomial de n, ou seja, é, na prática, um número pequeno. Tentativas de permitir que um número de desigualdades exponencialmente grande sejam candidatas à dualização Lagrangeana datam do início da década de 1980 (veja Balas e Christofides [5] e Gavish [48]). No entanto, esse objetivo só foi atingido de forma consistente através dos algoritmos *Relax and Cut*.

Originalmente, o termo *Relax and Cut* surgiu no paper de Escudero et al. [41] em um contexto diferente do que é empregado aqui. A explanação a seguir, baseada

no trabalho de Lucena [89], elucidará as diferenças.

Como em qualquer algoritmo baseado em Relaxação Lagrangeana, os algoritmos Relax-and-Cut são iniciados com a relaxação de um conjunto de restrições que complicam a resolução de uma dada formulação para um PI, enquanto restrições consideradas *fáceis* são preservadas. No algoritmo proposto em [41], procede-se à solução do PDL. Desigualdades válidas que violam a solução do PDL são então identificadas, podendo ser dualizadas ou preservadas (neste caso, podendo, ocasionalmente, modificar a estrutura dos PL(λ) envolvidos). Dualizadas ou preservadas as novas desigualdades, um novo PDL é formulado e resolvido. Este processo se repete até que algum critério de parada seja satisfeito. Observe que na abordagem de [41], a identificação de desigualdades violadas só ocorre quando o PDL é resolvido (de forma exata ou aproximada).

Já no algoritmo proposto por Lucena [86, 87], a identificação de desigualdades violadas *não é adiada* até que o PDL seja resolvido. Diferentemente de [41], algumas (poucas) desigualdades são dualizadas *on-the-fly*, isto é, na medida em que são violadas pela solução do PL(λ) corrente. Para enfatizar a diferença entre as duas abordagens, Lucena [89] denominou a abordagem empregada em [86, 87] de *Non-Delayed Relax-and-Cut* (NDRC) enquanto que aquela utilizada em [41] foi denominada de *Delayed Relax-and-Cut* (DRC).

Assumindo que os Problemas Duais Lagrangeanos sejam resolvidos na otimalidade, a convergência teórica dos algoritmos DRC (para um limite pelo menos tão bom quanto aquele obtido pela Relaxação Linear do problema considerado, reforçado pelas desigualdades usadas no algoritmo Relax-and-Cut) é garantida. Recentemente, Belloni e Sagastizabal [14] obtiveram um resultado similar para uma implementação de um algoritmo NDRC que emprega o Método de Bundle.

Para que as desigualdades dualizadas dinamicamente em um algoritmo Relax-and-Cut sejam capazes de elevar os limites z_d (independente do tipo de algoritmo empregado, DRC ou NDRC) em relação àqueles limites que seriam obtidos na ausência de sua dualização, é necessário que as mesmas não sejam simples combinações lineares das desigualdades explicitamente dualizadas no momento da solução de PDL. Este assunto é tratado em detalhes por Guignard [61].

Nesta Tese, trabalhamos com uma adaptação do MS para algoritmos NDRC. Embora a convergência teórica destes algoritmos não tenha sido ainda provada, resultados da literatura [12, 13, 16, 32, 86, 87, 96] indicam que o esquema tem um

bom desempenho na prática.

Detalhes da implementação do MS para algoritmos NDRC

Considere as desigualdades $\{a_i x \leq b_i, i = 1, \ldots, m\}$ dualizadas na k-ésima iteração do Método do Subgradiente, descrita no Algoritmo 2.1 da Seção 2.3. Estas desigualdades podem ser classificadas em três conjuntos, denominados respectivamente:

1. $CA(k)$: desigualdades ativas na iteração k, isto é, desigualdades violadas por $\overline{x}^k$. Note que tais desigualdades têm subgradientes $\{s_i^k = a_i\overline{x} - b_i\}$ estritamente positivos.

2. $PA(k)$: desigualdades anteriormente ativas, isto é, desigualdades cujos multiplicadores de Lagrange são positivos. Note que, se $\lambda_i^k > 0$, a desigualdade indexada por i foi necessariamente violada na solução de algum PL(λ^l), para $l < k$. Note também que os conjuntos $CA(k)$ e $PA(k)$ não são necessariamente disjuntos.

3. $CI(k)$: desigualdades inativas na iteração k, que correspondem a todas as demais desigualdades da formulação.

Para o caso de algoritmos baseados em Relaxação Lagrangeana Clássica, Beasley [10] destacou boas propriedades de convergência do MS, ao fixar $s_i^k = 0$, quando, simultaneamente, $s_i^k < 0$ (a desigualdade indexada por i não for violada por $\overline{x}^k$) e $\lambda_i^k = 0$. No contexto dos algoritmos NDRC, as desigualdades candidatas a esta modificação no MS são aquelas em $CI(k)$. Independente do fato de lidarmos com um número exponencialmente grande de desigualdades candidatas à dualização, seguimos a modificação sugerida em [10]. Isto porque, indiferentes às modificações sugeridas, os multiplicadores das desigualdades em $CI(k)$ não teriam seus valores (nulos) alterados, da iteração k para a iteração $k+1$. Além disto, as desigualdades em $CI(k)$ não contribuem diretamente para os custos Lagrangeanos na iteração k, mas, por outro lado, contribuem fortemente na determinação do tamanho de passo t^k. Tipicamente, para o caso das aplicações discutidas nesta Tese, o número de desigualdades em $CI(k)$ tende a ser muito grande. Assim sendo, se os subgradientes de todas as desigualdades em $CI(k)$ fossem explicitamente empregados no método, o

valor do tamanho de passo t^k seria muitíssimo pequeno, levando os multiplicadores a ficarem praticamente inalterados, iteração após iteração.

Em suma, seguindo a recomendação de Beasley [10], somos capazes de lidar adequadamente, dentro de um algoritmo Lagrangeano, com um número muito grande de desigualdades candidatas à dualização. Apesar disto, podemos ainda ter problemas com um número excessivamente grande de desigualdades em $CA(k) \setminus PA(k)$, isto é, com um número excessivo de desigualdades violadas por $\overline{x}^k$, na iteração k do método. Uma maneira de contornar esta dificuldade é a seguinte. Geralmente, estas desigualdades podem ser agrupadas em conjuntos, de acordo com algum critério dependente do problema a ser resolvido. De acordo com tal critério, para cada conjunto resultante, uma desigualdade de máxima violação deve existir. Tentando evitar que muitas variáveis sejam repetidamente penalizadas (através dos multiplicadores de Lagrange), adotamos a estratégia de dualizar a desigualdade mais violada de cada um desses conjuntos de desigualdades. As desigualdades remanescentes em $CA(k) \setminus PA(k)$ permanecem com seus multiplicadores (arbitrariamente) em 0 e portanto, na prática, passam a atuar, dentro da dinâmica do algoritmo, como as desigualdades encontradas em $CI(k)$.

Para finalizar esta Seção, cabe estabelecer um paralelo entre algoritmos NDRC e algoritmos poliedrais de Planos de Corte. Os algoritmos NDRC, na implementação descrita aqui (em conjunto com o MS), podem ser vistos como algoritmos tradicionais de Relaxação Lagrangeana nos quais um número exponencialmente grande de desigualdades são dualizadas, sendo seus subgradientes projetados, a cada iteração do MS, no espaço induzido pelas desigualdades ativas na iteração corrente. Assim sendo, para atualizar os multiplicadores de Lagrange, apenas uma diminuta fração das desigualdades são explicitamente consideradas, em cada iteração. Uma idéia análoga, consiste em tentar obter a Relaxação Linear de uma formulação contendo um número exponencialmente grande de desigualdades. Tipicamente, apenas uma fração destas desigualdades são ativas para uma dada solução da Relaxação Linear. Além disto, em um algoritmo de Planos de Corte, *poucas* desigualdades são efetivamente necessárias para se alcançar os limites duais por ele oferecidos. Assim sendo, os algoritmos NDRC podem ser considerados, em um contexto Lagrangeano, como análogos aos algoritmos de Planos de Corte em Relaxação Linear.

2.5 Algoritmos *Branch and Bound*

Genericamente, os algoritmos *Branch and Bound* consistem em um procedimento *inteligente* de enumeração de todas as possíveis soluções de um problema de Otimização Inteira ou Combinatória. Ao invés de enumerar explicitamente todas as possíveis soluções do problema, utiliza-se a combinação de duas estratégias principais: a primeira consiste em particionar o domínio do problema em uma série de subespaços disjuntos e a segunda consiste em encontrar limites duais para os problemas definidos em cada subespaço. À primeira estratégia dá-se o nome de *Branching* (ou *ramificação*) e à segunda, *Bounding* (ou *poda*).

O termo Branch and Bound, originalmente criado por Little et. al [84], refere-se a toda uma classe de algoritmos que originaram do trabalho de Croes [26] e Land e Doig [78], por volta da década de 1960.

Seguindo motivações puramente didáticas, vamos considerar a partir de agora, nesta Seção, que o PI a ser resolvido é um Problema de Otimização Inteira definido em varíáveis binárias $0-1$. Como poderá ser percebido ao longo do texto, isto não implica em perda de generalidade do método.

Para simplificar a notação empregada, vamos considerar que o PI a ser resolvido é descrito como

$$z = \min \left\{ c^t x : Ax \leq b, x \in \mathbb{B}^n \right\}, \tag{2.9}$$

onde $A \in \mathbb{Q}^{m \times n}, b \in \mathbb{Q}^m, c \in \mathbb{R}^n, \mathbb{B}^n = \{0,1\}^n$, e é denotado por PI_0.

Suponha que dispomos de um limite inferior z_0 e de um limite superior $\overline{z}_0$ válidos para PI_0, tais que $z_0 < \overline{z}_0$. Caso tais limites sejam os melhores que o nosso conhecimento atual do problema nos permite obter, recorremos então a uma forma de enumeração implícita das soluções do mesmo. Esta enumeração é feita da seguinte forma: ao invés de tentar resolver PI_0, vamos resolver, por exemplo, dois novos problemas PI_1 e PI_2, definidos respectivamente em espaços idealmente disjuntos. Suponha que, por um critério qualquer (*ramificação*), tenhamos escolhido uma variável x_j para gerar o particionamento. Então, resolver PI_0 equivale a resolver, em separado, os problemas PI_1 e PI_2, que se originam de PI_0, definidos respectivamente por:

$$\min \left\{ c^t x : Ax \leq b, x_j \leq 0, x \in \mathbb{B}^n \right\} \tag{2.10}$$

e

$$\min\left\{c^t x : Ax \leq b, x_j \geq 1, x \in \mathbb{B}^n\right\}, \tag{2.11}$$

e escolher, entre as duas soluções obtidas, aquela de menor custo.

Note que se as variáveis de definição do problema assumissem valores inteiros não necessariamente $0-1$ (como no caso tratado nesta Seção), o particionamento de PI_0 poderia ser feito, por exemplo, através das desigualdades $\{x_j \leq u_j\}$ e $\{x_j \geq u_j + 1\}$ para $u_j \in \mathbb{Z}$. Este tipo de particionamento é denominado *ramificação em variáveis*.

Note também que, para que o esquema faça sentido, resolver PI_1 e PI_2 deve ser mais fácil que resolver PI_0. Se isto não ocorre, devemos aplicar o mesmo procedimento a PI_1 e PI_2, e assim sucessivamente.

Observe que em função do particionamento de PI_0 em dois subproblemas PI_1 e PI_2, desenvolve-se uma *árvore de enumeração* binária. Cada nó i da árvore corresponde a um subproblema PI_i. A cada um deles, limites primais e duais podem ser associados.

Considere agora que na investigação do problema PI_1 tenhamos obtido, por um método qualquer, um limite inferior z_1 para PI_1 tal que $z_1 > \overline{z}_0$. Naturalmente, nesse caso, não é necessário prosseguir com o particionamento de PI_1, já que $z_1 > \overline{z}_0$ comprova não existir na região de viabilidade associada a PI_1 uma solução ótima para PI_0. Note também que, no caso em que $z_1 = \overline{z}_0$, poderíamos também excluir de nossa análise o espaço de soluções associado a PI_1. Isso se aplica pois o valor de qualquer solução viável em PI_1 teria um valor igual ou superior a $\overline{z}_0$.

Nos restaria agora prosseguir na investigação do *nó* da árvore de enumeração que ainda não foi investigado, correspondente ao problema PI_2. Suponha que ao investigá-lo, tenhamos obtido novos limites z_2 e $\overline{z}_2$ tais que $z_2 < \overline{z}_2 \leq \overline{z}_0$. Estes limites não permitem cessar a investigação do problema PI_0 nesse ramo da árvore, de forma que devemos particionar PI_2, por exemplo, em dois novos subproblemas disjuntos, tendo agora $\overline{z}_2$ como um limite superior global atualizado para PI_0. O procedimento de *ramificação* e *poda* prossegue até que, para todos os nós da árvore de enumeração, tenhamos a garantia de que um limite inferior obtido naquele nó supera ou iguala o melhor limite superior conhecido para o problema, ou que não é possível gerar uma solução viável a partir daquele nó (*poda por inviabilidade*).

É crucial destacar a generalidade do algoritmo *Branch and Bound*. Note que o método não se restringe ao critério de ramificação (em *variáveis*) descrito aqui. Note também que em momento algum particularizamos o modo como os limites

inferiores e superiores são obtidos. Dependendo do modo como são gerados estes limites, famílias distintas de algoritmos são obtidos. Uma destas famílias é tratada a seguir.

2.6 Algoritmos *Branch-and-Cut*

Os algoritmos Branch-and-Cut são algoritmos de Planos de Corte inseridos em um esquema de enumeração implícita Branch and Bound. Nestes algoritmos, os limites inferiores z_i em cada nó da árvore de enumeração são obtidos, inicialmente, a partir da Relaxação Linear da formulação inicial do subproblema PI_i. Se $\overline{x}$ é uma solução ótima, fracionária, para PI_i, antes de prosseguir com a estratégia de ramificação, tentamos fortalecer os limites inferiores no nó i, através da identificação de desigualdades válidas violadas por $\overline{x}$. Estas desigualdades são então incorporadas à formulação do problema PI_i e um novo limite inferior, reforçado, é obtido para o problema, através da Relaxação Linear correspondente.

Conforme destacado em Wolsey [125], embora a diferença entre um algoritmo Branch-and-Cut e um algoritmo Branch-and-Bound pareça pequena, ela implica em uma nova filosofia. Nos algoritmos Branch-and-Cut, tipicamente, investe-se mais tempo em cada nó da árvore de enumeração, seja para a identificação de desigualdades violadas, em rotinas de pre-processamento ou para obtenção de limites primais. Naturalmente, uma boa dosagem do esforço dispendido em cada nó antes de sua ramificação torna-se necessário para o bom andamento do algoritmo. Nem sempre, na prática, encontrar este equilíbrio é uma tarefa fácil.

Apesar do termo *Branch-and-Cut* ter sido proposto por Padberg e Rinaldi [108], o primeiro algoritmo deste tipo é creditado, por alguns pesquisadores, a Grötschel et al. [57]. Via de regra, algoritmos Branch-and-Cut contam com uma série de *ingredientes essenciais*, fundamentais para uma boa performance. Dentre estes ingredientes, podemos citar:

- Um conjunto de procedimentos de separação, capazes de identificar desigualdades violadas (preferencialmente facetas) pertencentes a uma ou mais famílias de desigualdades, escolhendo, dentre elas, aquelas que devem ser inseridas na Relaxação Linear corrente do problema.

- Uma estrutura denominada *cut-pool*, sugerida em [108], que armazena, convenientemente, desigualdades válidas para o problema. O cut pool possui duas

utilidades principais: facilitar a reconstrução das matrizes dos subproblemas ao longo da árvore de enumeração e permitir que desigualdades válidas sejam usadas em várias etapas do algoritmo, sem a necessidade de recorrer novamente a algoritmos de separação.

- Um conjunto de heurísticas que permitam obter limites superiores globais para o problema. Via de regra, informações presentes nas soluções das Relaxações Lineares dos subproblemas são usadas para guiar estas heurísticas.

- Um conjunto de rotinas de pré-processamento, que, idealmente, permitem reduzir o tamanho das instâncias antes do início da árvore de enumeração.

Além dos ingredientes acima, os algoritmos Branch-and-Cut possuem um sistema de regras que os controlam e integram. Estas regras visam, por exemplo, estabelecer os critérios para ramificação de nós (*regra de ramificação*) e prioridades para selecionar e explorar nós ainda não explorados (*regra de seleção de nós*), isto é, nós associados a subproblemas PI_i tais que $z_i < \overline{z}_i$.

O progresso de um algoritmo de Planos de Corte na tarefa de resolver um PI_i é normalmente medido pelo crescimento do correspondente limite z_i, na medida em que a formulação de PI_i é fortalecida com cortes. Tipicamente, os limites duais crescem rapidamente nas primeiras iterações de um algoritmo de Planos de Corte mas, a partir de um certo ponto, passam a crescer muito pouco. A esta *estagnação* dos limites duais, damos o nome de *tailing off*. Via de regra, quando detectado um tailing off, é mais vantajoso optar pela ramificação do que continuar tentando reforçar os limites inferiores no nó em questão, através de planos de corte.

Diversas técnicas têm sido empregadas na literatura para lidar adequadamente com o tailing off. Em Koch e Martin [75], foram utilizados cortes ortogonais em uma tentativa de evitar sua ocorrência em um algoritmo de Planos de Corte para o Problema de Steiner em Grafos. Técnica similar foi responsável por substanciais reduções do tempo de computação de um algoritmo de Planos de Corte implementado por Lucena e Resende [92] para o Problema de Steiner em Grafos com Recolha de Prêmios. Já em Padberg e Rinaldi [108], a ocorrência de tailing off é identificada ao se observar um número de iterações nas quais o aumento do limite dual é inferior a um parâmetro que, em [108], dependia das dimensões das instâncias consideradas. Uma regra similar mas um pouco mais simples, sugerida em [38], é empregada nesta Tese. Estabelecemos um número máximo (MAXITER) de Programas Lineares (ou

iterações) que são resolvidos consecutivamente em um mesmo nó da árvore, sem que um incremento mínimo MINIMP nos limites inferiores tenha ocorrido. Se, por mais de MAXITER iterações, o crescimento do limite dual for inferior a MINIMP, a execução do algoritmo de Planos de Corte é interrompida e uma ramificação de PI_i é então processada.

Um outro aspecto relevante à eficiência dos algoritmos Branch-and-Cut diz respeito à manutenção de restrições nas matrizes dos subproblemas e de cortes no cut-pool. Na medida em que muitas desigualdades são identificadas e inseridas no cutpool e/ou na matriz de um subproblema, torna-se fundamental adotar alguma política que, periodicamente, permita que algumas delas sejam eliminadas. Uma estratégia comum, adotada aqui, consiste em eliminar restrições não ativas da matriz dos subproblemas, sempre que houver crescimento do limite dual no nó em questão. As políticas de eliminação de desigualdades do cut-pool devem considerar, entre outros aspectos, o tempo gasto na separação de cada classe de desigualdades e o fato das desigualdades serem ou não globalmente válidas [108].

Capítulo 3

Um estudo poliedral do Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices

Neste capítulo, inicialmente, apresentamos uma Formulação de Programação Inteira para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices. Em seguida, apresentamos desigualdades válidas para o problema que fortalecem a formulação clássica. Finalmente, apresentamos um estudo das dimensões das desigualdades apresentadas com relação ao politopo do k-PARG, um caso particular do politopo associado ao PARG.

3.1 Introdução

Seja $G = (V, E)$ um grafo não direcionado, onde V é o conjunto de vértices e E é o conjunto de arestas. *Custos* $\{c_e \in \mathbb{Z} : e \in E\}$ são associados às arestas de E e limites superiores $\{1 \leq b_i \leq |V| - 1 : i \in V\}$ são associados aos *graus* permitidos para os vértices de V. Uma árvore geradora de G é denominada *viável* se, para qualquer vértice $i \in V$, o grau de i, naquela árvore, não excede b_i. O PARG consiste em encontrar uma árvore geradora viável de menor custo.

O PARG surge em várias aplicações práticas, como por exemplo, no projeto de redes de computadores [47] e de circutos elétricos [102], em sistemas de transportes nos quais há uma limitação no número máximo de entroncamentos entre vias, em sistemas hidráulicos, etc.

O k-PARG é um caso especial bastante conhecido do PARG, no qual o mesmo limite superior k é imposto a todo vértice em V. Se $k = |V| - 1$, o k-PARG equivale ao Problema da Árvore Geradora de Custo Mínimo (PAGM), para o qual algorit-

mos polinomiais são conhecidos [77, 111]. Se $k = 2$, o problema é equivalente ao de determinar o *Caminho Hamiltoniano* de mínimo custo de G, sendo, portanto $\mathcal{NP}$-completo [46]. É simples encontrar uma transformação polinomial do 2-PARG para o k-PARG (veja Papadimitriou e Vazirani [110]), para o caso onde os custos são arbitrários e $3 \leq k \leq n-2$. Desta forma, o k-PARG é ainda $\mathcal{NP}-$completo. Transformações similares não puderam ser obtidas para o caso em que os custos obedecem a uma métrica, porém Papadimitriou e Vazirani [110] mostraram que se os vértices do grafo correspondem a pontos no plano Euclidiano e se os custos das arestas correspondem à distância Euclidiana entre eles (sem aplicar arredondamento inteiro), o 3-PARG Euclidiano também é $\mathcal{NP}$-completo. Baseando-se na desigualdade triangular, os autores também mostraram que, se $k \geq 5$ e se as coordenadas dos vértices são inteiras, o k-PARG Euclidiano é polinomialmente solúvel. Conjectura-se que o 4-PARG Euclidiano seja $\mathcal{NP}-$completo [73]. As variantes do PARG nas quais restrições de grau (na forma de igualdade [44, 51] ou na forma de desigualdade [51, 126]) são aplicadas a um único vértice do grafo e, portanto, as solucões viáveis correspondem à interseção de dois matróides, são outros exemplos de casos polinomialmente solúveis.

Até o presente, nenhum trabalho tratou do estudo do PARG de um ponto de vista poliedral. Neste capítulo esperamos contribuir com a apresentação de desigualdades válidas para o PARG obtidas a partir de politopos de problemas correlatos. O Capítulo é organizado da seguinte forma: na Seção 3.2 apresentamos uma Formulação de Programação Inteira clássica para o problema. Nas Subseções 3.3.1 e 3.3.2, fortalecemos esta formulação clássica através da apresentação de desigualdades válidas para o problema, obtidas a partir do politopo do b-matching e do Caixeiro Viajante simétrico, respectivamente. Na Seção 3.4, estudamos a dimensão de algumas das desigualdades apresentadas com relação ao politopo do k-PARG.

3.2 Uma formulação de Programação Linear Inteira para o PARG

Associe variáveis binárias $\{x_e \in \{0,1\} : e \in E\}$ às arestas de G e denote por $E(S)$ o conjunto de arestas com ambas as extremidades em $S \subseteq V$. Adicionalmente, denote por $\delta(S)$ o conjunto de arestas com apenas uma extemidade em S. Similarmente, denote por $x(M)$ a soma das variáveis de decisão associadas às arestas de $M \subseteq E$.

Uma formulação clássica para o PARG é dada por

$$z = \min \left\{ \sum_{e \in E} c_e x_e : x \in P_1(n,b) \cap \mathbb{Z}^{|E|} \right\}, \tag{3.1}$$

onde $P_1(n,b)$ é a região poliedral definida por:

$$x(E) = |V| - 1, \tag{3.2}$$

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

$$x(\delta(i)) \leq b_i, \, \forall \, i \, \in \, V, \tag{3.4}$$

$$x_e \geq 0, \, \forall e \, \in \, E. \tag{3.5}$$

Note que os limites superiores do tipo $x_e \leq 1$, $\forall e \in E$, estão contidos em (3.3).

As desigualdades (3.3), denominadas *desigualdades de eliminação de subrotas* [30] (SECs), garantem que a solução não envolve ciclos. Já as desigualdades *b-Matching* (3.4) restringem as árvores admissíveis àquelas que satisfazem as restrições de grau.

Edmonds [36] provou que as desigualdades (3.2), (3.3) e (3.5) descrevem completamente o poliedro $AGM(n) := \text{conv}\{x \in \{0,1\}^{|E|} : x \text{ é uma árvore geradora de } G\}$.

A formulação (3.1) ou mesmo formulações mais fracas que (3.1), nas quais as desigualdades *cutsets*

$$x(\delta(S)) \geq 1, \;\; \forall S \subset V, \tag{3.6}$$

são empregadas em substituição às desigualdades de eliminação de subrotas (3.3), foram a base de todos os algoritmos disponíveis na literatura até a divulgação do presente estudo. Na próxima seção, apresentamos desigualdades válidas para o PARG que fortalecem (3.1).

3.3 Desigualdades válidas para o PARG

3.3.1 Desigualdades obtidas a partir do politopo dos b-matching

Dado um grafo não direcionado $G = (V, E)$, no qual são definidas capacidades para os vértices $\{b_i \in \mathbb{N}_+, \forall i \in V\}$ e *capacidades para as arestas* $\{u_e \in \mathbb{Z}_+, \forall e \in E\}$, um *b-matching* u-capacitado de G é um conjunto de arestas, contendo, eventualmente, mais de uma cópia de algumas delas, que satisfaz:

1. para todo vértice $i \in V$ existem no máximo b_i arestas incidentes em i;

2. para toda aresta $e \in E$ há no máximo u_e cópias da aresta e no conjunto.

Se, a cada aresta $e \in E$, associarmos uma variável de decisão x_e que represente o número de vezes que a aresta e é utilizada, então os vetores de incidência dos *b-matching* u-capacitados devem satisfazer ao seguinte sistema:

$$x(\delta(i)) \leq b_i \,, \forall\, i \,\in\, V, \tag{3.7}$$

$$0 \leq x_e \leq u_e, \; \forall e \,\in\, E, \tag{3.8}$$

$$x_e \,\in \mathbb{Z}_+, \; \forall e \in E. \tag{3.9}$$

Alguns casos particulares do sistema acima merecem ser destacados. Se $u_e = 1$, $\forall e \in E$, temos os *simple b-matchings* e, caso $x(\delta(i)) = b_i \,, \forall i \,\in\, V$, temos então um *perfect b-matching* u-capacitado.

Considere agora o politopo

$$CM(n, b, u) := \text{conv}\{x \in \mathbb{Z}_+^{|E|} \,:\, x \text{ é um b-matching u-capacitado de } G\,\},$$

isto é, a envoltória convexa do conjunto de vetores de incidência dos b-matchings u-capacitados. Edmonds [35, 37] mostrou que as desigualdades *Blossom*

$$x(E(H)) + x(T) \leq \left\lfloor \frac{1}{2}(b(H) + u(T)) \right\rfloor \,, \forall H \subseteq V, \; \forall T \subseteq \delta(H), \tag{3.10}$$

onde $b(H) := \sum_{i \in H} b_i$ e $u(T) := \sum_{e \in T \subseteq \delta(H)} u_e$ são válidas para o poliedro $CM(n, b, u)$. Além de provar a validade de (3.10), Edmonds também provou que $CM(n, b, u)$ é completamente caracterizado pelas desigualdades (3.7)-(3.8) e (3.10).

Informalmente, dizemos que conjuntos $H \subseteq V$ e $T \subseteq \delta(H) \subseteq E$ (usualmente denominados de *handle* e *teeth*, respectivamente) induzem uma desigualdade Blossom. Estamos particularmente interessados em pares H, T para os quais a soma $b(H) + u(T)$ seja ímpar, embora a validade das desigualdades Blossom não se restrinja a esta condição.

Conforme estabelecemos formalmente no resultado enunciado a seguir, as desigualdades Blossom são também válidas para o PARG.

Proposição 3.1 *As desigualdades Blossom*

$$x(E(H)) + x(T) \leq \frac{b(H) + |T| - 1}{2}$$

para $b(H) + |T|$ ímpar, são válidas para o PARG.

Prova: Considere a soma das $|H|$ desigualdades em (3.7) e a soma das $|T|$ arestas em (3.8), ou seja: $2x(E(H))+x(\delta(H)) \leq b(H)$ e $x(T) \leq |T|$, respectivamente. Some estas duas desigualdades e obtenha $2x(E(H)) + x(\delta(H)) + x(T) \leq b(H) + |T|$. Já que $x_e \geq 0$, temos que $2x(E(H)) + 2x(T) \leq b(H)) + |T|$. Agora divida por dois e aplique arredondamento inteiro, obtendo: $x(E(H)) + x(T) \leq \left\lfloor \frac{1}{2}(b(H) + |T|) \right\rfloor$. Como $b(H) + |T|$ é ímpar, o resultado segue. ■

Observe que uma solução viável para o PARG é um *b-matching 1*-capacitado com topologia de árvore. Em uma perspectiva mais ampla, o conjunto das árvores geradoras de G que satisfazem as restrições de grau correspondem aos vértices comuns de dois politopos inteiros: o politopo $AGM(n)$ das árvores geradoras [36] e o politopo $CM(n, b, \mathbb{1})$ (onde $\mathbb{1} := (1, \ldots, 1)^t$ denota um vetor $|E|$ dimensional de 1's) dos b-matching 1-capacitados [35, 37]. Seja então $P_2(n, b) := AGM(n) \cap CM(n, b, \mathbb{1})$, a interseção destes dois poliedros inteiros. Se definirmos $PARG(n, b) := \text{conv}\{x \in \mathbb{B}^{|E|} : x \in P_1(n, b)\}$ (o politopo associado à envoltória convexa dos vetores característicos das árvores geradoras de G que satisfazem às restriões de grau), temos que $PARG(n, b) \subseteq P_2(n, b)$. Como veremos na Seção a seguir, o conjunto de vértices do poliedro $P_2(n, b)$ pode ter pontos extremos distintos daqueles presentes em um dos dois politopos originais. De fato, são conhecidos poucos casos na literatura nos quais a interseção de dois politopos inteiros permanece inteira (veja [45, 64], por exemplo).

Para encerrar esta Seção, identificamos um caso que ilustra que $P_2(n, b) \subset P_1(n, b)$. Na Figura 3.1, logo a seguir, indicamos um ponto extremo $\overline{x} \in P_1(n, b)$ tal que $\overline{x} \notin CM(n, b, \mathbb{1})$ para o 4-PARG. Note que a desigualdade Blossom induzida pelos conjuntos $H = \{1, 2, 3\}$ e $T = \{(1, 5), (1, 6), (1, 12), (3, 4), (3, 8), (3, 11), (2, 7), (2, 9), (2, 10)\}$ é violada em $\frac{1}{2}$, já que $\overline{x}(E(H)) = 1\frac{1}{2}$, $\overline{x}(T) = 9$, $b(H) = 12, |T| = 9$. Não é difícil verificar, por inspeção, que não há SECs violadas por $\overline{x}$.

3.3.2 Desigualdades generalizadas a partir do Problema do Caixeiro Viajante

Informalmente, o Problema do Caixeiro Viajante (PCV) consiste em se identificar uma permutação dos n vértices de V que induza um *ciclo Hamiltoniano* de mínimo

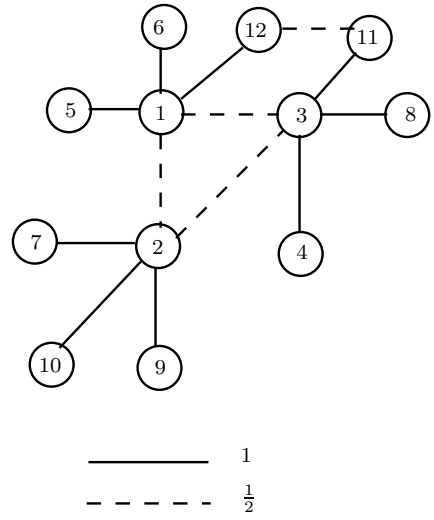

Figura 3.1: Um ponto extremo $\overline{x} \in P_1(n,b), \overline{x} \notin CM(n,b,\mathbb{1})$ para o 4-PARG.

custo. Um ciclo Hamiltoniano é um caminho em G que inicia e termina em um mesmo vértice, no qual todos os demais são visitados uma única vez. Matematicamente, o PCV pode ser formulado como

$$\min \left\{ \sum_{e \in E} c_e x_e : x \in H_1(n) \cap \mathbb{Z}^{|E|} \right\}, \tag{3.11}$$

onde $H_1(n)$ é a região poliedral definida por:

$$x(E) = |V|, \tag{3.12}$$

$$x(E(S)) \leq |S| - 1, \, \forall S \subset V, \, S \neq \emptyset, \tag{3.13}$$

$$x(\delta(i)) = 2, \, \forall \, i \, \in \, V, \tag{3.14}$$

$$x_e \geq 0, \, \forall e \, \in \, E. \tag{3.15}$$

Como mostraremos a seguir, algumas desigualdades válidas para o politopo $PCV(n) := \text{conv}\{x \in \mathbb{B}^{|E|} \, : \, x \text{ induz um ciclo Hamiltoniano de } G\}$, possuem contrapartidas que são válidas para $PARG(n,b)$. A primeira destas desigualdades consideradas aqui são as *Combs*, propostas para o PCV por Grötschel e Padberg [58]. Trataremos também de uma generalização das Combs, denominadas *Clique-Trees*, que foram originalmente propostas por Grötschel e Pulleyblank [60].

3.3.2.1 Desigualdades Combs

Definição 3.1 *Sejam $H, T_j, j = 1, ..., k$ conjuntos de vértices de V. Dizemos que $H, T_1, ..., T_k$ induzem uma Comb para o PARG, se as seguintes condições são satisfeitas:*

1. $|H \cap T_i| \geq 1,\ \forall i \in \{1, 2, ..., k\}$;

2. $|T_i \setminus H| \geq 1,\ \forall i \in \{1, 2, ..., k\}$;

3. $|T_i \cap T_j| = \emptyset, \forall i, j \in \{1, 2, ..., k\}, i \neq j$;

4. $b(H) + k$ *é ímpar.*

De forma análoga às desigualdades Blossom, o conjunto H é denominado *handle* e os conjuntos T_j (agora definidos em termos de vértices e não mais de arestas) são chamados de *teeth* da desigualdade Comb.

Proposição 3.2 *As desigualdades Comb induzidas por* $H, T_j : j = 1, ..., k$

$$x(E(H)) + \sum_{i=1}^{k} x(E(T_i)) \leq \sum_{i=1}^{k} |T_i| + \left(\frac{b(H) - 3k - 1}{2} \right), \tag{3.16}$$

que satisfazem às condições da Definição 3.1, são válidas para PARG.

Prova. A prova da validade das Combs para o PARG segue aquela apresentada em Nemhauser e Wolsey [104] para o caso do PCV. Some as desigualdades (3.4) para todos os vértices em H e multiplique por $\frac{1}{2}$, obtendo

$$x(E(H)) + \frac{1}{2} x(\delta(H)) \leq \frac{b(H)}{2}.$$

Uma vez que $x_e \geq 0, \forall e \in E$, segue que

$$x(E(H)) + \frac{1}{2} \sum_{i=1}^{k} \sum_{e \in \delta(H) \cap E(T_i)} x_e \leq \frac{b(H)}{2}. \tag{3.17}$$

Agora multiplique as SECs

$$x(E(T_i)) \leq |T_i| - 1,\ i = 1, ...k,$$

$$x(E(T_i \cap H)) \leq |T_i \cap H| - 1,\ i = 1, ...k,$$

$$x(E(H \setminus T_i)) \leq |H \setminus T_i| - 1,\ i = 1, ...k,$$

por $\frac{1}{2}$ e some com (3.17), para obter

$$x(E(H)) + \tfrac{1}{2} \sum_{i=1}^{k} \left\{ \sum_{e \in \delta(H) \cap E(T_i)} x_e + \sum_{e \in E(T_i)} x_e + \sum_{e \in E(H \cap T_i)} x_e + \sum_{e \in E(T_i \setminus H)} x_e \right\} \leq$$
$$\tfrac{1}{2} \left(b(H) + \sum_{i=1}^{k} (|T_i| + |H \cap T_i| + |T_i \setminus H|) - 3k \right).$$

Uma vez que

$$\sum_{e\in\delta(H)\cap E(T_i)} x_e + \sum_{e\in E(H\cap T_i)} x_e + \sum_{e\in E(T_i\setminus H)} x_e = \sum_{e\in E(T_i)} x_e$$

e

$$|H\cap T_i| + |T_i\setminus H| = |T_i|,$$

temos que

$$x(E(H)) + \sum_{i=1}^{k}\sum_{e\in E(T_i)} x_e \leq \sum_{i=1}^{k} |T_i| + \frac{1}{2}(b(H) - 3k). \tag{3.18}$$

Como o lado esquerdo de (3.18) é inteiro e $b(H) + k$ é ímpar, podemos substituir $\frac{1}{2}(b(H) - 3k)$ por $\lfloor\frac{1}{2}(b(H) - 3k)\rfloor = \frac{b(H)-3k-1}{2}$ e o resultado segue. ∎

Note que a dedução da validade das desigualdades Comb é similar à dedução das desigualdades Blossom. Na verdade, ambas são $\{0, \frac{1}{2}\}$-cortes de Chvátal-Gomory [19] que diferem pela introdução de desigualdades SECs (3.3) no sistema linear que gera o corte de Chvátal-Gomory. Note também que uma desigualdade Comb pode ter coeficiente 2 e que as Combs generalizam as Blossom na medida em que um tooth pode induzir uma clique com mais de uma aresta.

Considere agora o poliedro $P_3(n, b) := \{x \in \mathbb{R}^{|E|} : x \in P_2(n, b) \cap (3.16)\}$ e o ponto extremo $\overline{x}$ indicado na Figura 3.2. Repare que $\overline{x}$ viola a Comb induzida pelos conjuntos $H = \{1, ..., 6\}$, $T_1 = \{4, 9, 10\}$, $T_2 = \{5, 7\}$ e $T_3 = \{6, 8\}$ para $b := [2, 1, 2, 3, 2, 2, 2, 2, 2, 2, 3, 1]$. Note que $b(H) + k$ é ímpar e que o lado direito de (3.16) vale 8 enquanto o lado esquerdo soma $8\frac{1}{2}$. Não existem desigualdades Blossom violadas por $\overline{x}$, isto é, $\overline{x} \in P_2(12, b)$. A certificação de inexistência de desigualdades Blossom violadas por $\overline{x}$ foi feita com o algoritmo exato de Letchford et al. [82] (veja o Apêndice B).

A seguir tratamos de uma desigualdade ainda mais geral que as Combs, as Clique-Trees.

3.3.2.2 Desigualdades Clique-Trees

A validade das desigualdades Clique-Trees com relação ao politopo $PCV(n)$ foi estabelecida por Grötschel e Pulleyblank [60]. Estas desigualdades são definidas em termos de *handles* $H_i, i = 1, ..., r$, *teeth* $T_j, j = 1, ..., s$ e de parâmetros $t_j, j = 1, ..., s$ que determinam o número de handles que cada tooth T_j intercepta.

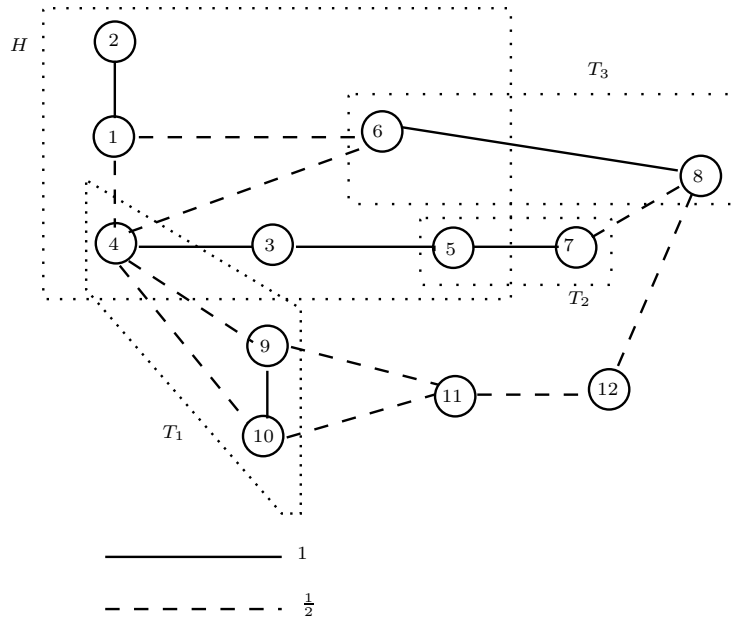

Figura 3.2: Um ponto extremo $\overline{x} \in P_2(12, b)$ que viola uma desigualdade Comb.

Definição 3.2 *Para o PARG, dizemos que H_i, T_j, t_j, r, s induzem uma Clique-Tree*

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_i)) \leq \sum_{j=1}^{s} (|T_j - t_j|) + \frac{1}{2}\left(\sum_{i=1}^{r} b(H_i) - s - 1\right) \qquad (3.19)$$

se as seguintes condições forem satisfeitas:

1. $T_i \cap T_j = \emptyset, i \neq j$;
2. $H_i \cap H_j = \emptyset, i \neq j$; ;
3. *Cada tooth possui pelo menos um e no máximo $|V| - 2$ vértices e pelo menos um vértice não pertencendo a nenhum handle;*
4. *Para cada handle H_k e os teeth $T_1, ..., T_p$ que o interceptam, $b(H_k) + p$ é ímpar.*

As Clique-Trees recebem este nome porque o grafo induzido pela desigualdade assume a topologia de uma árvore, se cada clique definida pelos conjuntos H_i e T_j for aglutinada em um único vértice e se todas as arestas com extremidades nos mesmos pares de Cliques também forem aglutinadas em uma única aresta. A Figura 3.3 ilustra os conjuntos que induzem uma desigualdade Clique-Tree. Nesta Figura, elipses indicam teeth enquanto círculos indicam handles. Note que a Figura indica a existência, em cada tooth, de um vértice (um círculo colorido em preto) que não pertence a nenhum handle e que, neste caso, $r = 4$, $s = 10$.

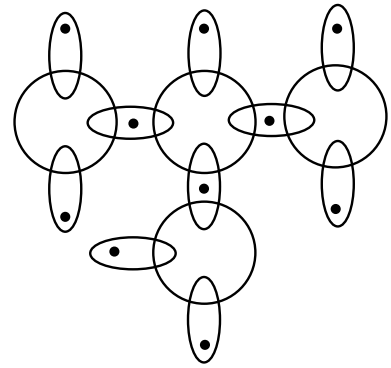

Figura 3.3: Representação dos conjuntos de uma Clique-Tree.

A demonstração da validade das Clique Trees para o PARG é idêntica à demonstração de validade das Clique-Trees para o PCV, apresentada originalmente por Grötschel e Pulleyblank [60]. Apesar dos argumentos de prova empregados aqui não serem originais, consideramos pertinente a apresentação dos mesmos porque permitirá identificar por quais motivos é necessária a imposição das condições estabelecidas na Definição 3.2. Antes de estabelecê-la formalmente, apresentamos algumas definições e resultados auxiliares, análogos aos apresentados para o PCV em [60].

Definição 3.3 *O* **tamanho** $sz(C)$ *de uma Clique Tree* C *induzida por* $H_1, ..., H_r, T_1, ..., T_s, t_1, ..., t_s, s, r$ *é dado por*

$$sz(C) = \sum_{j=1}^{s}(|T_j| - t_j) + \frac{1}{2}\left(\sum_{i=1}^{r} b(H_i) - s - 1\right).$$

Observe que o tamanho de uma Clique-Tree corresponde ao lado direito de (3.19).

Definição 3.4 *Seja* C *uma desigualdade Clique Tree e* $H \subset V$ *e* $T \subset V$*, respectivamente, um handle e um tooth de* C *que se interceptam. A operação de* **separação** *de* C **no tooth e no handle** *consiste em:*

1. *Remover todos os vértices em* $H \setminus T$ *de* C*. Seja* C_2 *o componente conexo de* $C \setminus (H \setminus T)$ *que contém* T*.*

2. *Remover todos os vértices de* C *que estão em handles que interceptam* T*, mas que não estão em* T *ou em* H*. Seja* C_1 *o componente conexo do grafo resultante que contém* T*.*

A Figura 3.4 ilustra a operação de separação (da Clique Tree indicada na Figura 3.3) no tooth T e no handle H. Note que C_1 e C_2, resultantes da operação, são Clique-Trees (as condições dadas pela Definição 3.2 são preservadas).

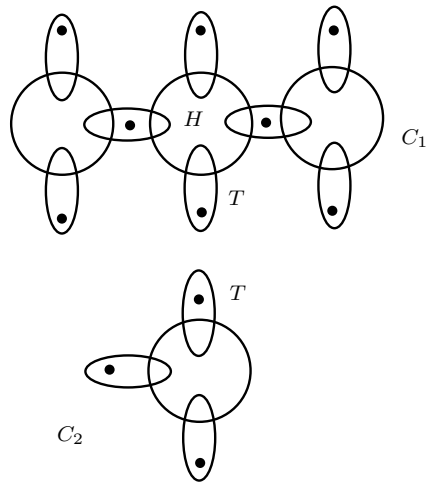

Figura 3.4: A operação de separação no tooth T e no handle H.

Definição 3.5 *Seja H um handle e $T_1, T_2, ..., T_k$ os teeth que o interceptam na Clique Tree C. Para cada tooth $T_i, i \in \{1, ..., k\}$, seja C_i a correspondente Clique Tree que não contém H, que seria obtida a partir de C pela operação de separação em T_i e H. As Clique Trees $C_1, ..., C_k$ são obtidas através da operação de* **separação** *de C* **no handle** *H.*

A Figura 3.5 ilustra a operação de separação da Clique Tree no handle.

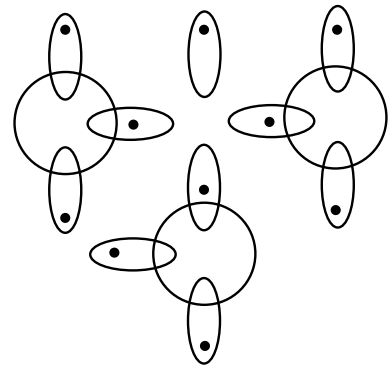

Figura 3.5: A operação de separação no handle.

Definição 3.6 *Seja $s(C_i)$ o número de teeth da Clique Tree C_i.*

Nos resultados que seguem, admitimos o abuso de notação $H_i \in C_y$ ($T_j \in C_y$), que significa que H_i (T_j) é um handle (tooth) da Clique Tree C_y. O Lema a seguir

indica a relação entre o lado direito de (3.19) para a Clique Tree original e as Clique Trees resultantes da operação de separação no tooth e no handle.

Lema 3.1 *(Análogo à Observação 3.6b em [60]) Sejam C_1 e C_2 duas Clique-Trees obtidas a partir de C pela operação de separação de C no tooth T e no handle H. Então:*

$$sz(C_1) + sz(C_2) = sz(C) + |T| - 1.$$

Prova: Temos que:

1. $\sum_{H_i \in C_1} b(H_i) + \sum_{H_i \in C_2} b(H_i) = \sum_{i=1}^{r} b(H_i)$, já que não há handle em comum entre C_1 e C_2;

2. $\sum_{T_j \in C_1} |T_j| + \sum_{T_j \in C_2} |T_j| = \sum_{i=1}^{r} |T_j| + |T|$, já que T é o único tooth em comum entre C_1 e C_2;

3. $s(C_1) + s(C_2) = s + 1$;

4. $\sum_{T_j \in C_1} t_j + \sum_{T_j \in C_2} t_j = \sum_{i=1}^{r} t_j$, já que o conjunto de handles que interceptam T em C_1 são C_2 disjuntos.

Somando o tamanho das Clique-Trees C_1 and C_2 e usando as expressões acima, temos:

$$\begin{aligned}
sz(C_1) + sz(C_2) &= \\
\sum_{T_j \in C_1} |T_j| + \sum_{T_j \in C_2} |T_j| &- \\
\left(\sum_{T_j \in C_1} t_j + \sum_{T_j \in C_2} t_j \right) &+ \\
\tfrac{1}{2} \left(\sum_{H_i \in C_1} b(H_i) + \sum_{H_i \in C_2} b(H_i) - s(C_1) - s(C_2) - 2 \right) &= \\
\sum_{j=1}^{r} (|T_j| - t_j) + \frac{1}{2} \left(\sum_{H_i \in C_1} b(H_i) + \sum_{H_i \in C_2} b(H_i) - s(C_1) - s(C_2) \right) + |T| - 1 &= \\
sz(C) + |T| - 1.
\end{aligned}$$

■

Já o próximo Lema indica como o tamanho da Clique Tree original se relaciona à soma dos tamanhos das Clique Trees obtidas com a operação de separação no handle.

Lema 3.2 *(Análogo à Observação 3.6c em [60]) Seja C uma Clique-Tree e H um de seus handles, que é interceptado por k teeth: $T_1, T_2, ..., T_k$. Sejam $C_1, C_2, ..., C_k$ as Clique-Trees obtidas pela operação de separação de C em H. Então:*

$$\sum_{y=1}^{k} sz(C_y) = sz(C) - \frac{b(H)}{2} + \frac{k+1}{2}.$$

Prova: Temos que:

1. $\sum_{y=1}^{k} \sum_{H_i \in C_y} b(H_i) + b(H) = \sum_{i=1}^{r} b(H_i)$, já que H é o único handle que não está na união de $C_1, ..., C_k$;
2. $\sum_{y=1}^{k} s(C_y) = s(C) = s$;
3. $\sum_{y=1}^{k} \sum_{j=1}^{s(C_y)} |T_j| = \sum_{j=1}^{s} |T_j|$;
4. $\sum_{y=1}^{k} \sum_{j=1}^{s(C_y)} t'_j + k = \sum_{j=1}^{s} t_j$, onde t'_j é o número de handles que interceptam o tooth $T_j \in C_y$. Isto se aplica já que para cada Clique-Tree $C_y, y = 1, ..., k$, exatamente um tooth perdeu um handle que o interceptava na operação de separação em H.

Então temos:

$$\begin{aligned}
\sum_{y=1}^{k} sz(C_y) &= \sum_{y=1}^{k} \left[\sum_{j=1}^{s(C_y)} (|T_j| - t'_j) + \frac{1}{2} \left(\sum_{H_i \in C_y} b(H_i) - s(C_y) - 1 \right) \right] \\
&= \sum_{y=1}^{k} \sum_{j=1}^{s(C_y)} |T_j| - \sum_{y=1}^{k} \sum_{j=1}^{s(C_y)} t'_j + \frac{1}{2} \sum_{y=1}^{k} \left(\sum_{H_i \in C_y} b(H_i) - s(C_y) - 1 \right) \\
&= \sum_{j=1}^{s} |T_j| - \left(\sum_{i=1}^{s} t_j - k \right) + \frac{1}{2} \left(\sum_{i=1}^{r} b(H_i) - b(H) - s(C) - k \right) \\
&= \sum_{j=1}^{s} (|T_j| - t_j) + \frac{1}{2} \left(\sum_{i=1}^{r} b(H_i) - s(C) - 1 \right) + \frac{1}{2} - \frac{1}{2}k + k - \frac{1}{2}b(H) \\
&= sz(C) - \tfrac{1}{2}b(H) + \tfrac{k+1}{2}.
\end{aligned}$$

Finalmente, podemos apresentar o resultado principal desta Seção, a Proposição a seguir.

Proposição 3.3 *(Análogo ao Teorema 3.7 em [60]) As desigualdades Clique-Trees (3.19) são válidas para $PARG(n, b)$.*

Prova: A prova é baseada em indução no número de *handles*. Se a Clique Tree não possui handles (caso base), existe apenas um tooth e então a Clique Tree resume-se a uma SEC, cuja validade é conhecida. Caso contrário suponha que (hipótese de indução) as Clique Trees são válidas para r (ou menos) handles e assuma que C é uma Clique Tree com handles $H_1, H_2, ..., , H_r, H_{r+1}$. Sejam $T_1, T_2, ..., T_k$ os teeth que interceptam o handle H_{r+1} e $C_1, C_2, ..., C_k$ be as Clique Trees obtidas pela operação de separação de C em H_{r+1}. Considere ainda que C_i contenha T_i, $i \in \{1, 2, ..., k\}$ e que $s = \sum_{i=1}^{k} s(C_i)$.

Cada uma das $C_1, ..., C_k$ Clique Trees possui no máximo r handles e pela hipótese de indução induzem uma desigualdade Clique-Tree válida. Então, seja $a_i x \leq sz(C_i)$ a desigualdade associada a C_i. Para cada uma, seja $\bar{C}_i$ a Clique Tree obtida após remover de T_i os vértices em $T_i \cap H_{r+1}$ (ou substituir T_i por $(T_i \setminus H_{r+1})$. Analogamente, seja $\bar{a}_i\ x \leq sz(\bar{C}_i)$ a desigualdade associada a $\bar{C}_i$. Note que a condição 4 da Definição 3.2 garante que $\bar{C}_i$ é uma Clique-Tree. Pelo Lema 3.2 temos

$$\sum_{i=1}^{k} sz(\bar{C}_i) = sz(C) - \frac{1}{2}b(H_{r+1}) + \frac{k+1}{2} - \sum_{k=1}^{r} |H_{r+1} \cap T_i|.$$

Então temos que:

$$\begin{aligned}
2\left(\sum_{i=1}^{r+1} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j))\right) &\leq \\
\sum_{i=1}^{k} \left(a_i x + \bar{a}_i\ x + x(E(H_{r+1} \cap T_i))\right) + \sum_{v \in H_{r+1}} x(\delta(v)) &\leq \\
\sum_{i=1}^{k} \Big(sz(C_i) + sz(\bar{C}_i) + |H_{r+1} \cap T_i| - 1\Big) + b(H_{r+1}) &= \\
2sz(C) - b(H_{r+1}) + k + 1 - k + b(H_{r+1}) &= \\
2sz(C) + 1 &
\end{aligned}$$

Já que para todas as soluções viáveis do PARG o lado esquerdo da desigualdade acima deve ser inteiro, se dividirmos por 2 e tomarmos a parte inteira, a prova fica completa. ■

Na Figura 3.6, indicamos o grafo induzido por um ponto extremo $\overline{x} \in P_2(20, b)$ que viola a Clique-Tree obtida quando $b = [2, 2, 1, 2, 2, 2, 3, 1, 3, 3, 2, 1, 2, 3, 2, 2, 2, 1, 3, 2]$, $s = 5$, $r = 2$, $H_1 = \{2, 3, 4, 5, 6, 7\}$, $H_2 = \{11, 12, 13, 14, 15, 16\}$, $T_1 = \{4, 7, 10, 11, 14\}$, $T_2 = \{13, 17\}$, $T_3 = \{16, 18\}$, $T_4 = \{5, 6, 8, 9\}$, $T_5 = \{1, 2\}$. Note que o tamanho da Clique-Tree é 18 enquanto o lado esquerdo de (3.19) soma 19. Nesta Figura, as desigualdades Combs definidas

para $H_2, T_6 = \{10, 11, 14\}, T_2, T_3$ e $H_1, T_7 = \{5, 6, 9\}, T_5, T_8 = \{4, 10\}$ também são violadas por $\overline{x}$ em $\frac{1}{2}$. Introduzindo estas Combs na formulação e reotimizando, obtemos uma nova solução fracionária $\overline{x} \in P_2(20, b)$, indicada na Figura 3.7. Note também que a desigualdade Clique-Tree acima permanece violada (por $\frac{3}{4}$) pela nova solução $\overline{x}$. Pelo que conhecemos, esta solução $\overline{x} \in P_2(20, b)$ não viola nenhuma Comb. Introduzindo agora esta Clique-Tree na formulação e reotimizando, obtemos uma solução fracionária final $\overline{x}$ indicada na Figura 3.8.

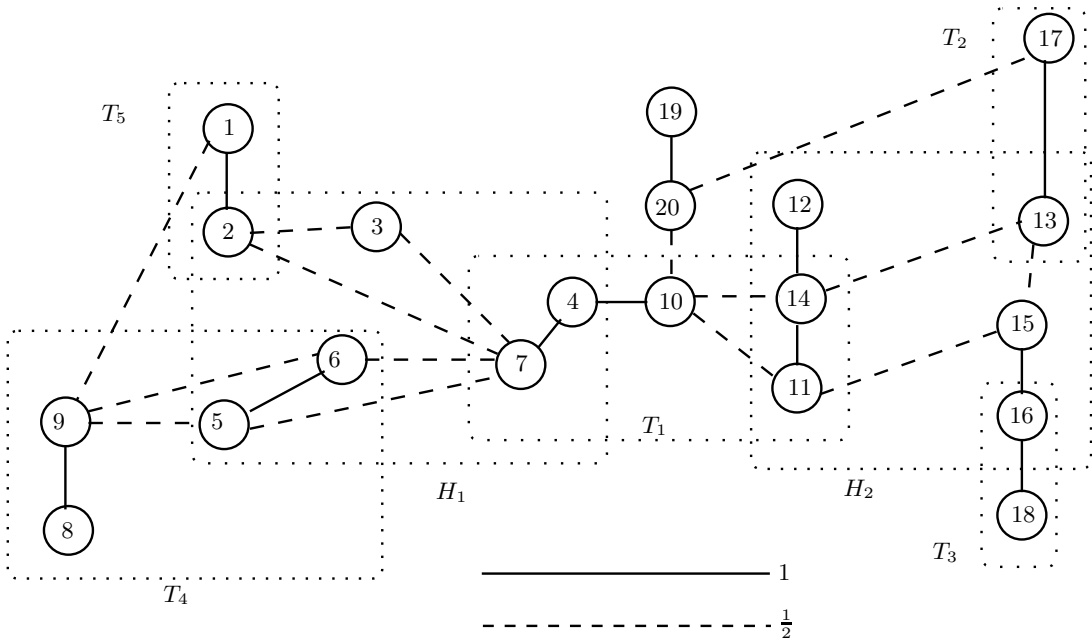

Figura 3.6: Desigualdades Clique-Trees e Combs violadas por $\overline{x} \in P_2(20, b)$

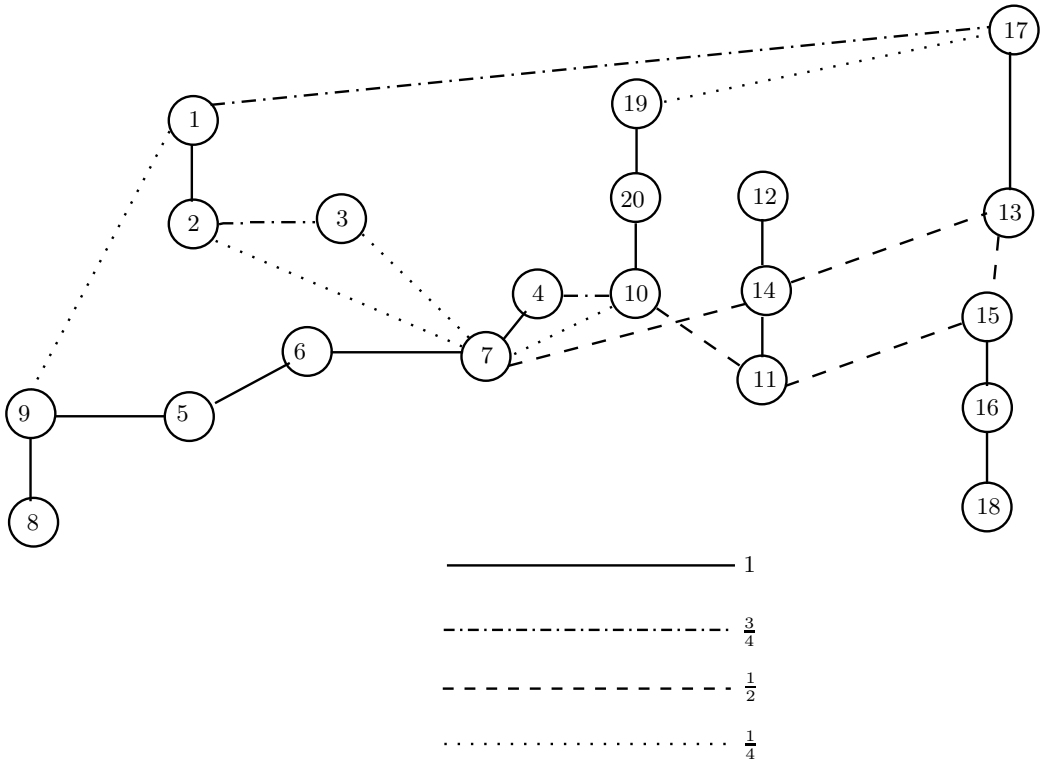

Figura 3.7: A desigualdade Clique-Tree ainda é violada por uma segunda solução $\overline{x} \in P_2(20, b)$.

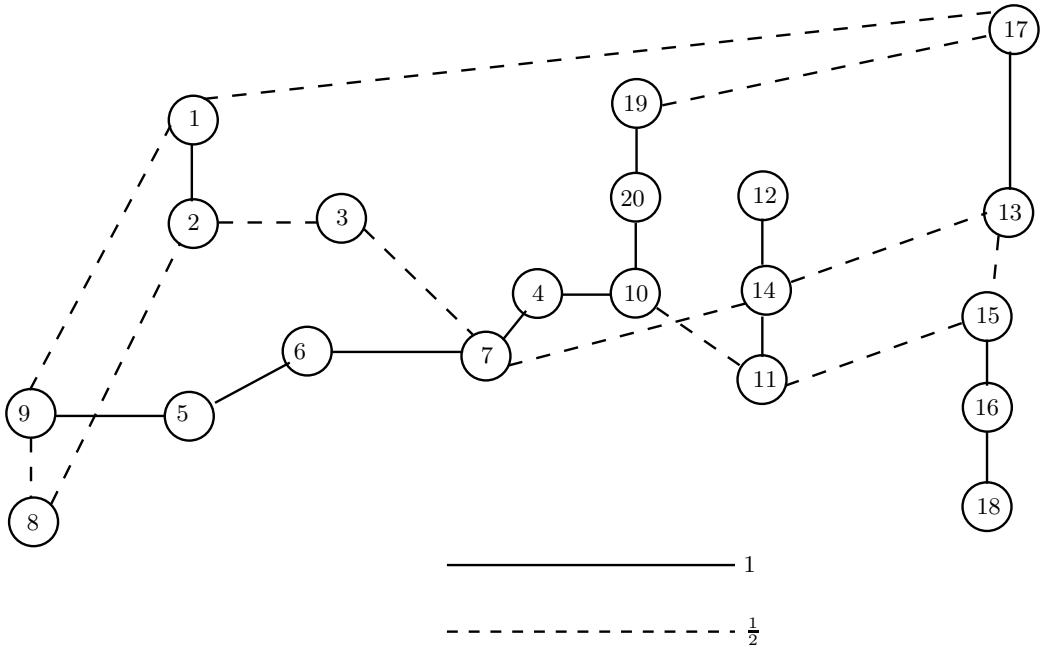

Figura 3.8: Solução fracionária final $\overline{x} \in P_2(20, b)$

3.4 O estudo das dimensões de desigualdades válidas para o k-PARG

Nesta Seção, trataremos do estudo das dimensões de desigualdades válidas para o caso particular $kPARG(n)$ de $PARG(n,b)$, para o qual $b_i = k \geq 3, \forall i \in V$. Ao longo do texto, faremos observações de como as propriedades discutidas são (ou não) preservadas para casos mais gerais ($b_i \geq 2$ ou mesmo $b_i = 1$ para alguns vértices). A razão principal para não considerarmos vértices i com $b_i = 1$ no estudo poliedral é que a sua introdução dificulta bastante o estudo das dimensões das faces, uma vez que a dimensão do politopo deixa de ser *quase-completa*. Fixando $b_i = k$, as provas apresentadas tornam-se mais simples e não precisamos assumir quaisquer hipóteses sobre distribuição dos limites de grau nos vértices envolvidos nas desigualdades. No estudo a seguir, assumimos que os grafos são completos.

Nos resultados que seguem, considere que $(a, a_0) \in \mathbb{R}^{|E|} \times \mathbb{R}$ (ou equivalentemente $\{x \in \mathbb{R}_+^{|E|} : ax \leq a_0\}$) denota uma desigualdade válida para $kPARG(n)$ e que $G(a) = (V, E(a))$ representa o grafo por ela induzido (isto é, $E(a) := \{e \in E : a_e \neq 0\}$). Dizemos que (a, a_0) é uma desigualdade *própria*, se a face induzida por (a, a_0) no politopo $kPARG(n)$ é própria. A seguir, apresentamos um resultado simples, sobre a caracterização de desigualdades próprias para $kPARG(n)$, que será usado ao longo desta Seção.

Lema 3.3 *Seja $(a, a_0) \in \mathbb{R}^{|E|} \times \mathbb{R}$, $a \neq 0$ uma desigualdade própria para $kPARG(n)$. Sem perda de generalidade, podemos assumir que (a, a_0) é escrita na forma (única)* **normalizada***:*

1. $a_e \geq 0, \forall e \in E$;
2. $min\{a_e : a_e \neq 0\} = 1$;
3. $a_e = 0$, *para alguma aresta* $e \in E$.

Prova: Suponha que (a, a_0) não esteja escrita na forma normalizada acima. Defina $\mu_1 = -\min\{a_e : a_e \neq 0\}$. Por hipótese, $a \neq 0$, e então μ_1 é bem definido para alguma aresta $f \in E$ para a qual o mínimo é obtido. Defina $(\hat{a}, \hat{a}_0)$ como a seguinte desigualdade $\{x \in \mathbb{R}_+^{|E|} : (\mu_1 \mathbb{1} + a)x \leq \mu_1(|V| - 1) + a_0\}$. Note que $\hat{a}_f = 0$ e $\hat{a}_e \geq 0, \forall e \in E$. Defina $\mu_2 = \min\{\hat{a}_e : \hat{a}_e > 0\}$ e faça $(\overline{a}, \overline{a}_0) = \frac{1}{\mu_2}(\hat{a}, \hat{a}_0)$. Note que μ_2 também é bem definido, digamos, para uma aresta $g \in E$. Após esta normalização,

temos $\overline{a}_f = 0$, $\overline{a}_e \geq 0, \forall e \in E$ e $\overline{a}_g = 1$. Assim sendo $(\overline{a}, \overline{a}_0)$ e (a, a_0) são equivalentes em relação ao poliedro $kPARG(n)$, isto é, induzem a mesma face de $kPARG(n)$.

■

À luz do Lema anterior, podemos assumir, para o estudo das faces induzidas por desigualdades válidas para o $kPARG(n)$, que qualquer desigualdade própria para este politopo apresenta-se na forma normalizada. Nas Seções seguintes, vamos denotar por $F(a, a_0) := \text{conv}\{x \in kPARG(n) : ax = a_0\}$ a face induzida em $kPARG(n)$, por uma desigualdade (a, a_0).

3.4.1 Dimensões das faces induzidas por desigualdades da formulação clássica

Para procedermos à análise das dimensões de $kPARG(n)$, faremos uso de alguns resultados preliminares, apresentados por Queyranne e Wang [112]. Naquela referência os autores estudaram os politopos do Caixeiro Viajante simétrico $PCV(n)$ e do *Caminho Hamiltoniano* $2PARG(n)$.

Para melhor compreender os resultados de [112], considere $\overline{G} = (\overline{V}, \overline{E})$, um grafo não direcionado onde $\overline{V} = V \cup \{n+1\}, \overline{E} = E \cup \{(i, n+1) : i = 1, ..., n\}$. Observe que o grafo completo G definido em n vértices pode ser obtido a partir de $\overline{G}$ eliminando-se o vértice $n+1$ e todas as arestas em $\overline{E}$ incidentes a ele. Considere x^{T_j}, o vetor de incidência de um tour em $\overline{G}$. Definimos a *projeção* de x^{T_j} no espaço $\mathbb{R}^{|E|}$ como o vetor x tal que $x_e = 1$ se $x_e^{T_j} = 1, e \in E$, $x_e = 0$, caso contrário. Observe que a projeção de x^{T_j} em $\mathbb{R}^{|E|}$ (ou equivalentemente, no subespaço gerado pelas arestas de G) define um caminho hamiltoniano. Mais formalmente, a projeção de $x^{T_j} \in \mathbb{R}^{|\overline{E}|}$ no subespaço $\mathbb{R}^{|E|}$ é o vetor de incidência de um caminho hamiltoniano x^{P_j}. Consequentemente, o poliedro $2PARG(n)$ é a projeção de $PCV(n+1)$ no subespaço $\mathbb{R}^{|E|}$.

Sejam então $x^{T_j}, j = 1, ..., p$ vetores de incidência de p tours $T^j, j = 1, ..., p$ em $\overline{G}$. Queyranne e Wang mostraram (Lema 2.1 em [112]) que se a matriz $[x^{T_1}, x^{T_2}, ..., x^{T_p}] \in \mathbb{B}^{|E'|} \times \mathbb{B}^p$ possui posto p ($p \leq |E'| - (n+1)$), então sua projeção em $\mathbb{R}^{|E|}$ também possui posto p. Este resultado projetivo permite estabelecer a equivalência poliédrica de uma desigualdade válida para a $PCV(n+1)$ e sua projeção em $\mathbb{R}^{|E|}$ com relação a $2PARG(n)$. Por um lado, a projeção de uma desigualdade (a, a_0), válida para $PCV(n+1)$, no subespaço $\mathbb{R}^{|E|}$ é válida para $2PARG(n)$. Por outro lado, a face induzida pela desigualdade (a, a_0) em $PCV(n+1)$ e a face indu-

zida pela sua projeção em $2PARG(n)$ possuem a mesma dimensão. Adicionalmente, Queyranne e Wang também mostraram os seguintes resultados:

Lema 3.4 *(Teorema 2.2 em [112]) A dimensão de* $2PARG(n)$ *é* $d = \frac{n(n-1)}{2} - 1$.

Lema 3.5 *(Proposição 2.9 em [112]). As desigualdades a seguir definem facetas para* $2PARG(n), n \geq 4$ *e nenhuma delas é equivalente com relação a* $2PARG(n)$*:*

1. $x_e \geq 0$, $\forall e \in E$;
2. $x(\delta(v)) \leq 2$, $\forall v \in V$;
3. $x(E(W)) \leq |W| - 1$, $\forall W : 2 \leq |W| \leq n - 1$.

O resultado a seguir é uma conseqüência direta dos Lemas 3.4 e 3.5.

Corolário 3.1 *Para* $kPARG(n)$ *com* $k \geq 3$ *e* $n \geq 4$ *podemos afirmar:*

1. *A dimensão (dim) de* $kPARG(n)$ *é* $d = \frac{n(n-1)}{2} - 1$;
2. *As desigualdades de não negatividade* $x_e \geq 0$, $\forall e \in E$ *e de eliminação de subrotas* $x(E(W)) \leq |W| - 1$, $\forall W : 2 \leq |W| \leq n - 1$ *definem facetas de* $kPARG(n)$.

Prova:

$2 - PARG(n) \subseteq kPARG(n)$ para $k \geq 3$. Logo $\dim(kPARG(n)) \geq \dim(2PARG(n))$. Como qualquer árvore geradora que satisfaz às restrições de grau deve satisfazer (3.2), temos que $\dim(kPARG(n)) \leq \frac{n(n-1)}{2} - 1$ e a primeira afirmativa segue.

Vamos provar que $x_e \geq 0$ define facetas de $kPARG(n)$ para $n \geq 4$. A prova para as desigualdades SECs é análoga. Pelo Lema 3.5, existe um caminho Hamiltoniano que utiliza a aresta e. Este caminho Hamiltonino é uma solução válida para $kPARG$, $k \geq 3$. Portanto, $x_e \geq 0$ induz uma face própria de $kPARG(n)$. Também pelo Lema 3.5, existem d caminhos hamiltonianos $P_j, j = 1, ..., d$ cujos vetores de incidência são afim independentes e que satisfazem $x_e^{P_j} = 0, j = 1, ..., d$. É óbvio que estes caminhos hamiltonianos são árvores válidas para $kPARG$ para $k \geq 3$. Logo a dimensão da face induzida pela desigualdade $x_e \geq 0$ é exatamente $d - 1$. ■

Observação 3.1 *Note que o Corolário 3.1 também é válido para* $PARG(n, b), b_i \geq 2$.

Observação 3.2 *Ao introduzirmos vértices com limites de grau unitários, reduzimos a dimensão de* $PARG(n, b)$, *já que* $x(\delta(i)) = 1$ *e* $x_{(i,j)} = 0$ *para qualquer árvore viável se* $b_i = b_j = 1$.

Observação 3.3 *Se* $b_i = 1$ *para algum* $i \in S \subset V$, $x(E(S)) \leq |S| - 1$ *é dominada pela soma de* $x(E(S \setminus \{i\})) \leq |S| - 2$ *e* $x(\delta(i)) \leq 1$. *Portanto, a SEC induzida por* S *não pode gerar uma faceta* $PARG(n, b)$.

Para prosseguir com o estudo das dimensões das desigualdades de grau (3.4) e Blossom (3.10), considere o seguinte resultado a seguir.

Proposição 3.4 *Seja* (a, a_0) *uma desigualdade própria para* $kPARG(n)$, $k \geq 3$ *escrita em sua forma normalizada. Seja* $G(a) = (V(a), E(a))$ *o grafo induzido por* (a, a_0). *Se existe uma desigualdade* $(d, d_0) \in \mathbb{R}^{|E|} \times \mathbb{R}$, $d \neq 0$ *que induz uma faceta* $F(d, d_0)$ *de* $PARG(n, b)$ *tal que* $F(a, a_0) \subseteq F(d, d_0)$ *e que possa ser escrita (na forma normalizada) como* $d_e = \alpha$, $\forall e \in E(a)$, $d_e = \beta$, $\forall e \in E \setminus E(a)$, *então* $F(a, a_0) = F(d, d_0)$, *isto é,* (a, a_0) *define uma faceta de* $kPARG(n)$.

Prova Pelo Lema 3.3, temos que pelo menos um coeficiente d_e deve ser nulo em (d, d_0). Considere então os seguintes casos possíveis:

1. $\alpha = 1, \beta = 0$. Neste caso, $(a, a_0) = (d, d_0)$.

2. $\alpha = 0, \beta = 1$. Tome $x^j \in F(a, a_0)$. Note que, por hipótese, $x^j \in F(d, d_0)$. Portanto temos: $\sum_{e \in E(a)} x_e^j = a_0$, $\sum_{e \in (E \setminus E(a))} x_e^j = d_0 = |V| - 1 - a_0$. Logo (d, d_0) pode ser escrita como $\sum_{e \in (E \setminus E(a))} x_e \leq |V| - 1 - a_0$. Agora subtraia $\sum_{e \in E} x_e = |V| - 1$ de $\sum_{e \in E(a)} x_e \leq a_0$ para obter $-\sum_{e \in (E \setminus E(a))} x_e \leq a_0 - |V| + 1$ ou $\sum_{e \in (E \setminus E(a))} x_e \geq |V| - 1 - a_0$. Diante disto, temos uma contradição, já que $\sum_{e \in (E \setminus E(a))} x_e = |V| - 1 - a_0$ e portanto $F(d, d_0)$ não é uma face própria de $kPARG(n)$.

Portanto, a face e a faceta coincidem. ∎

Estamos agora em condição de enunciar o seguinte:

Teorema 3.1 *As desigualdades de grau* $x(\delta(v)) \leq k$, $2 \leq k \leq n - 2$, *induzem facetas de* $kPARG(n)$ *para* $n \geq 4$.

Prova:

Vamos considerar o caso onde $k \geq 3$, já que o caso onde $k = 2$ foi provado em [112]. Sem perda de generalidade, vamos assumir que $v = k+1$. Assuma que (a, a_0) represente a desigualdade $x(\delta(k+1)) \leq k$. Não é difícil perceber que (a, a_0) induz uma face própria de $kPARG(n)$. Note que esta desigualdade já está em sua forma normalizada e que $E(a) = \delta(k+1)$. Seja $F(d, d_0)$ uma faceta (escrita em sua forma normalizada) que contenha a face induzida por (a, a_0).

Primeira parte: vamos mostrar que $d_e = \alpha, \forall e \in \delta(k+1)$. Considere as soluções x^1, x^2 indicadas na Figura 3.9. Note que x^1 e x^2 diferem apenas pelas arestas tracejadas, isto é, $(k, k+1)$ pertence a x^1 mas não pertence a x^2. Já que $dx^1 = d_0$ e $dx^2 = d_0$, podemos estabelecer que $d_{(k+1,k)} = d_{(k+1,n)}$ e, como $k \geq 3$, construção semelhante envolvendo os demais vértices $k+2, k+3, ..., n$ permite dizer que $d_{(k+1,i)} = d_{(k+1,j)},\ i, j \in \{k+2, ..., n\}$. Comparando agora as soluções x^3 e x^4 indicadas na Figura 3.10, podemos dizer que $d_{(k+1,1)} = d_{(k+1,n)}$. Repetindo este processo para $j = 2, ..., k-1$, temos que $d_{(k+1,j)} = d_{(k+1,n)}, j = 1, ..., k-1$. Portanto, $d_e = \alpha, \forall e \in \delta(k+1)$.

Segunda parte: vamos mostrar que $d_e = \beta$ para $e \in E \setminus E(a)$. Primeiro particionemos $E \setminus E(a) = E_1 \cup E_2 \cup E_3$ da seguinte forma:

- $E_1 = E(\{k+2, ..., n-1, n\})$;
- $E_2 = E(\{1, 2, ..., k\})$;
- $E_3 = \delta(i : j) = \{(i, j) \in E : i \in \{1, 2, ..., k\}, j \in \{k+2, ..., n-1, n\}\}$.

Considere as soluções x^5, x^6 indicadas na Figura 3.11. Podemos dizer que $d_{(n,n-2)} = d_{(n,n-1)}$. Considere também construções semelhantes a x^5, x^6, nas quais substituimos, em x^5, a aresta $(n, n-2)$ por $(n, j) : j \in \{k+2, ..., n-1\}$ e, em x^6, a aresta $(n, n-1)$ por $(i, i-1)$ para $i \in \{n-1, ..., j\}$. Está claro que $d_e = d_f = \beta_{E_1}$ para quaisquer $e, f \in E_1$.

Considere agora a floresta obtida pela união da *estrela* $\{(k+1, 1), ...(k+1, k)\}$ e do caminho hamiltoniano $\{(k+2, k+3), ..., (n-1, n)$. Conectando estes dois componentes por meio de qualquer aresta $(i, j), i \in \{1, ..., k\}, j \in \{k+2, ..., n\}$ (veja soluções x^7, x^8 na Figura 3.12) temos soluções que satisfazem a restrição de grau na igualdade. Comparando estas possíveis soluções, não é difícil concluir que $d_e = d_f = \beta_{E_3}, \forall e, f \in E_3$.

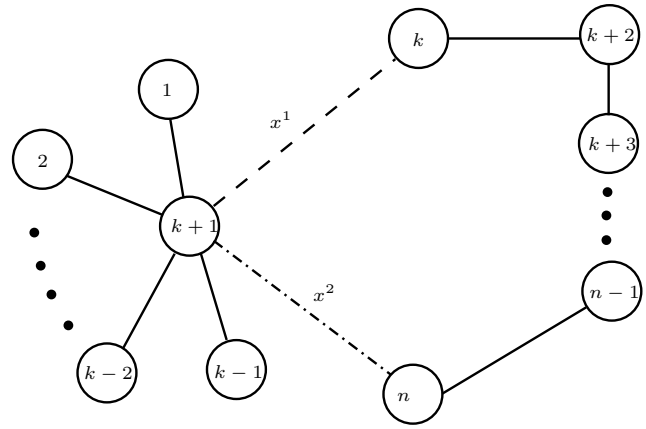

Figura 3.9: Soluções viáveis x^1, x^2 satisfazendo $x(\delta(k+1)) = k$

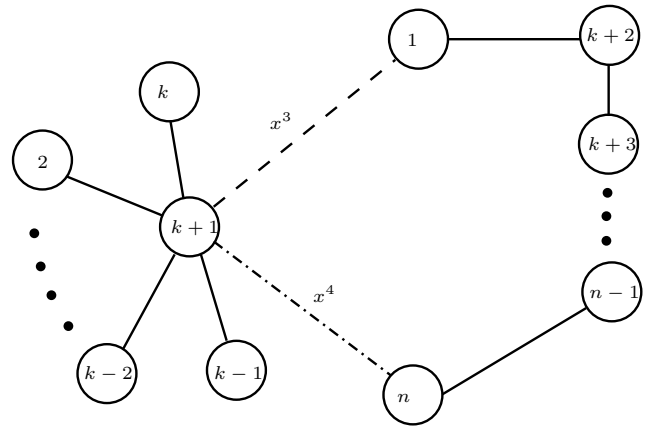

Figura 3.10: Soluções viáveis x^3, x^4 satisfazendo $x(\delta(k+1)) = k$

Tendo em mente que $d_e = \alpha, e \in E(a)$ e que $dx^9 = dx^{10}$ (veja Figura 3.13), através de construções similares a x^9, x^{10}, pode-se estabelecer que $d_e = d_f = \beta_{E_2}, e, f \in E_2$.

Finalmente, já que $dx^7 = dx^9$, temos que $\beta_{E_3} = d_{(k-1,n)} = d_{(1,k)} = \beta_{E_2} = \beta$. Considere novamente a união da estrela $\{(k+1,1), ...(k+1,k)\}$ e do caminho hamiltoniano $\{(k+2,k+3), ..., (n-1,n)$. Conecte-os por meio de $(k,k+2)$, obtendo x^{11}. Agora remova $(k+2,k+3)$ e reconecte a floresta através de $(k,k+3)$, obtendo x^{12}. Como $dx^{11} = dx^{12}$ temos que $\beta_{E_3} = d_{(k,k+2)} = d_{(k+2,k+3)} = \beta_{E_1}$ e a prova está completa, em conseqüência da Proposição 3.4. ■

3.4.2 Dimensão das faces induzidas pelas desigualdades Blossom

Antes de prosseguir com o estudo da dimensão das desigualdades Blossom para $kPARG(n)$, vamos estabelecer as condições necessárias para que a face por ela

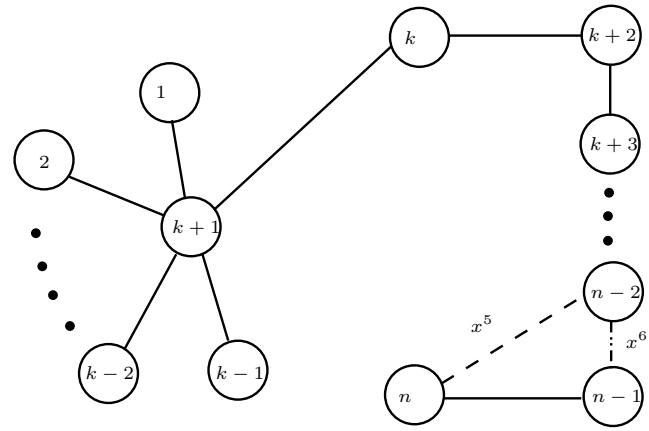

Figura 3.11: Soluções viáveis x^5, x^6 satisfazendo $x(\delta(k+1)) = k$

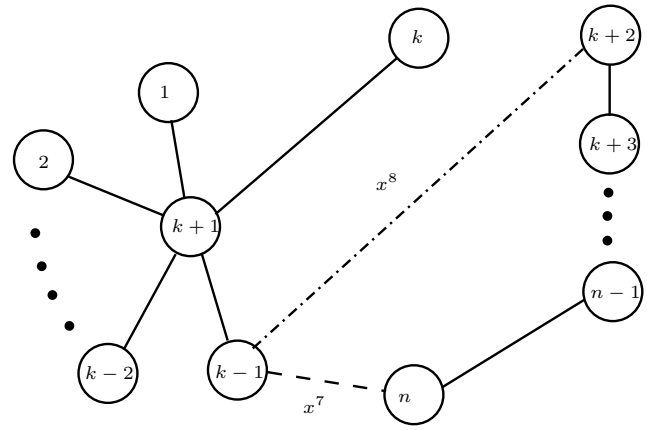

Figura 3.12: Soluções viáveis x^7, x^8, satisfazendo $x(\delta(k+1)) = k$

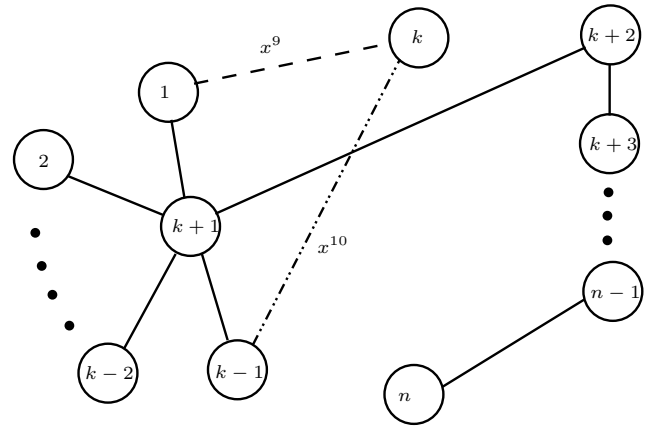

Figura 3.13: Soluções viáveis x^9, x^{10}, satisfazendo $x(\delta(k+1)) = k$

induzida seja própria. O Lema a seguir trata desta questão.

Lema 3.6 *Suponha que $H \subseteq V$ e $T \subseteq \delta(H) \subset E$ sejam tais que $b(H)+|T|$ é ímpar. A desigualdade blossom gerada (ou induzida) por H, T é satisfeita na igualdade por uma árvore x^i que satisfaz às restrições de grau, se:*

1. *ou $b(H) = 2x^i(E(H)) + x^i(\delta(H))$ e*

   (a) *ou exatamente uma aresta em T não é saturada (isto é, $x^i_e = 1, \forall e \in T \setminus \{f\}, x^i_f = 0, f \in T$) e todas as outras arestas em $\delta(H) \setminus T$ são não saturadas;*

   (b) *ou todas as arestas em T são saturadas e exatamente uma aresta em f em $\delta(H) \setminus T$ é também saturada ($x^i_f = 1, f \in \delta(H) \setminus T$).*

2. *ou $b(H) = 2x^i(E(H))+x^i(\delta(H))+1$ e $x_e = 1, \forall\, e \in T$ e $x_e = 0$, $\forall e \in \delta(H)\setminus T$.*

Prova: Considere a desigualdade blossom induzida por H, T escrita em sua forma (fechada) equivalente: $\sum_{e\in\delta(H)\setminus T} x_e + \sum_{e\in T}(1 - x_e) + \sum_{i\in H}(b_i - x(\delta(i)) \geq 1$. Vale lembrar que se a desigualdade é justa na forma (3.10) ela também é justa na forma fechada. Note que todos os termos da desigualdade são não negativos e, portanto, se $\sum_{i\in H}(b_i - x(\delta(i)) \geq 2$, a blossom não é justa. Como cada contribuição em cada somatório do lado esquerdo da desigualdade deve ser inteira para o vetor de incidência de qualquer árvore viável, podemos restringir $b(H)-2x^i(E(H))-x^i(\delta(H))$ a zero ou um.

Observamos também que não podemos ter mais de um termo com contribuição positiva na desigualdade acima. Assim se o handle é folgado, isto é, $b(H) - 2x^i(E(H)) - x^i(\delta(H)) = 1$, os demais termos devem ser nulos. Por outro lado, se o handle é justo, exatamente um dos demais deve contribuir com uma unidade para o lado esquerdo da desigualdade. ■

Este Lema facilitará a criação de operações de troca de arestas entre soluções que satisfazem desigualdades Blossom na igualdade. Em conjunto com a Proposição 3.4, estas operações são a base de prova do Teorema 3.2 que segue. Para enunciá-lo, indexamos as extremidades de arestas em T que não estão em H, a partir das respectivas extremidades em H. Isto é feito da seguinte forma: se i é um vértice em H, a partir de i existem exatamente $k-1$ arestas escolhidas em $T \subseteq \delta(H)$ que possuem i como extremidade. Então, as arestas em T incidentes a i são: $\{(i, i+zh) : z = 1, \ldots, k-1$ e $h = |H|\}$.

Teorema 3.2 *Seja* $H = \{1, ..., 3\}, h = |H| = 3, b_i = k \geq 3, \forall i \in H$. *Seja* $T = \{(i, i+zh) : i \in H,\ z = 1, ..., k-1\}$. *A desigualdade Blossom gerada por* H, T *induz uma faceta para* $kPARG(n)$, $n = 3k$.

Prova:

Primeiramente, note que $|T| = 3(k-1)$ e $V \setminus H = \{4, ..., 3k\}$ é o conjunto de vértices que são extremidades de arestas de T que não estão no handle H. Veja que por construção, $b(H) + |T| = 6k - 3$ é ímpar e que a desigualdade é própria. Note também que o lado direito de 3.10 vale $3k - 2$ e que portanto, qualquer solução válida que satisfaça a desigualdade Blossom na igualdade precisa de $3k - 2$ arestas saturadas em $E(a)$.

Sejam $F(a, a_0)$ a face induzida pela desigualdade Blossom (a, a_0) e $F(d, d_0)$ uma faceta induzida por uma desigualdade própria (d, d_0) para $kPARG(n)$, que contém a face $F(a, a_0)$.

A seguir, construiremos soluções viáveis que, de acordo com o Lema 3.6, satisfazem a desigualdade induzida por H, T na igualdade. Considere inicialmente as soluções viáveis em $F(a, a_0)$, $\tilde{x}^1$,$\tilde{x}^2$, dadas pela união das arestas em T, $(1, 3)$ e $(5, 3k)$ e $T \cup (1, 2) \cup (5, 3k)$, respectivamente, indicadas na Figura 3.14. Considere também as soluções $\tilde{x}^3, \tilde{x}^4$ indicadas na Figura 3.15, construídas analogamente. Já que $d\tilde{x}^1 = d\tilde{x}^2$ e $d\tilde{x}^3 = d\tilde{x}^4$, temos que $d_e = \alpha_{E(H)}, \forall e \in E(H)$.

Considere agora as soluçoes $\hat{x}^1, \hat{x}^2$ indicadas na Figura 3.16. Claramente, temos que $d_e = \alpha_{T_i},\ \forall e = (i, i+zh), i \in H, z \in \{1, ..., k-1\}$. Já que $d_e = \alpha_{E(H)}, \forall e \in E(H)$, construções $\hat{x}^3, \hat{x}^4$ similares às indicadas, respectivamente, nas Figuras 3.17 e 3.18, permitem estabelecer que $\alpha_{T_i} = \alpha_{T_j} = \alpha_T,\ i \neq j, i, j \in H$.

Vamos agora particionar $E \setminus E(a) = E_1 \cup E_2 \cup E_3$ da seguinte forma:

- $E_1 = \delta(H) \setminus T$.
- E_2: conjunto de arestas cujas extremidades pertencem a $V \setminus \{1, 2, 3\}$ e que não se conectam ao mesmo vértice i de H por arestas em T, isto é: $E_2 = \{(i + zh, j + ph) : i \neq j, i, j \in \{1, 2, 3\}, z, p \in \{1, ..., k-1\}\}$. Por exemplo, $(4, 6) \in E_2$.
- E_3: conjunto das demais arestas cujas extremidades pertencem a $V \setminus \{1, 2, 3\}$, isto é: $E_3 = \{(i+jh, i+zh), i \in \{1, 2, 3\}, j \neq z \in \{1, ..., k-1\}\}$. Por exemplo, $(4, 7) \in E_3$.

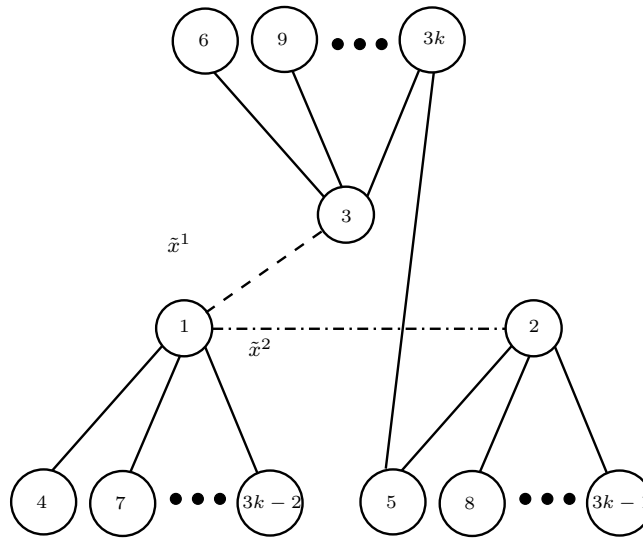

Figura 3.14: Soluções viáveis $\tilde{x}^1, \tilde{x}^2$ satisfazendo a desigualdade Blossom na igualdade

Considere então as construções x^1, x^2 (Figura 3.19) x^3, x^4 (Figura 3.20) e x^5, x^6 (Figura 3.21). Uma vez que $d_e = \alpha_{E(H)}, \forall e \in E(H)$ e que $dx^1 = dx^2 = dx^5$, construções análogas a x^1, x^2, x^5 (nas quais apenas uma aresta em $E(H)$ é escolhida) estabelecer que $d_e = d_f = \beta, \forall e, f \in E_2$. Comparando x^3 e x^4 (e construções similares nas quais uma das duas arestas em $E(H)$ é substituida por outra aresta em $E(H)$) podemos dizer que $d_e = d_f = \beta, \forall e, f \in E_2 \cup E_3$. Já que $dx^5 = dx^6$, construções análogas a x^5, x^6 (por exemplo a solução viável x^7 indicada na Figura 3.19) permitem estabelecer que $d_e = d_f = \beta, \forall e, f \in E \setminus E(a)$.

Por fim, considere as soluções $\tilde{x}^1$ (Figura 3.14) e $\hat{x}^2$ (Figura 3.16). Já que $d_{(5,3k)} = d_{(6,3k)}$, temos que $d_{(3,6)} = d_{(2,3)}$, isto é, $\alpha_{E(H)} = \alpha_T$, completando a prova ■.

Observação 3.4 *Se o número de arestas em T incidentes a um determinado vértice $i \notin H$ é igual a b_i a correspondente desigualdade Blossom não pode induzir uma faceta para $PARG(n, b)$ já que ela é dominada pela desigualdade Blossom que seria obtida eliminando-se de T estas arestas e introduzindo i em H.*

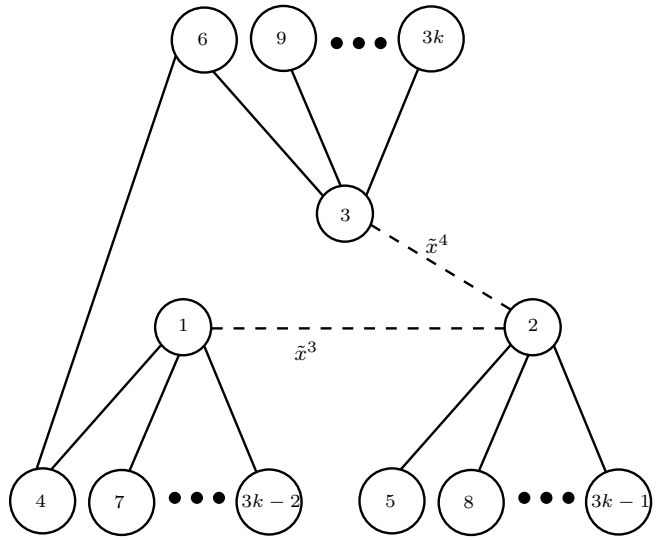

Figura 3.15: Soluções viáveis $\tilde{x}^3, \tilde{x}^4$ satisfazendo a desigualdade Blossom na igualdade

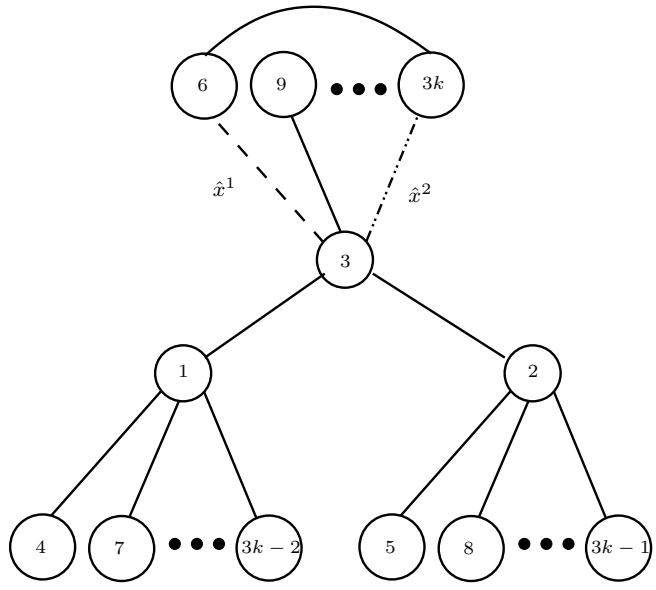

Figura 3.16: Soluções viáveis $\hat{x}^1, \hat{x}^2$ satisfazendo a desigualdade Blossom na igualdade

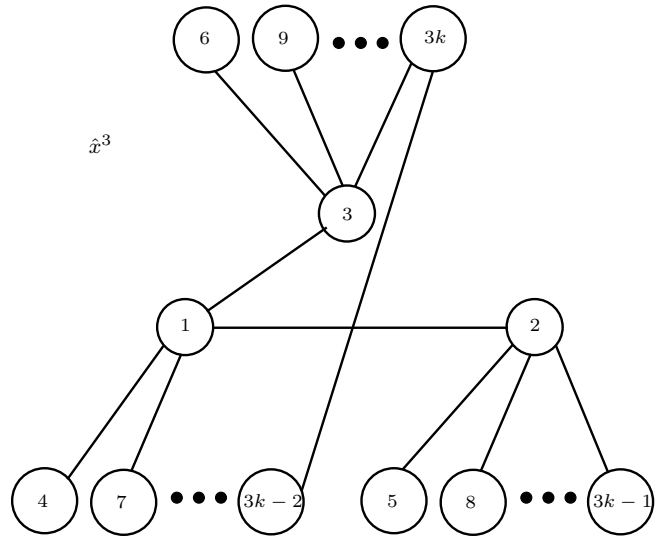

Figura 3.17: Solução viável $\tilde{x}^3$ satisfazendo a desigualdade Blossom na igualdade

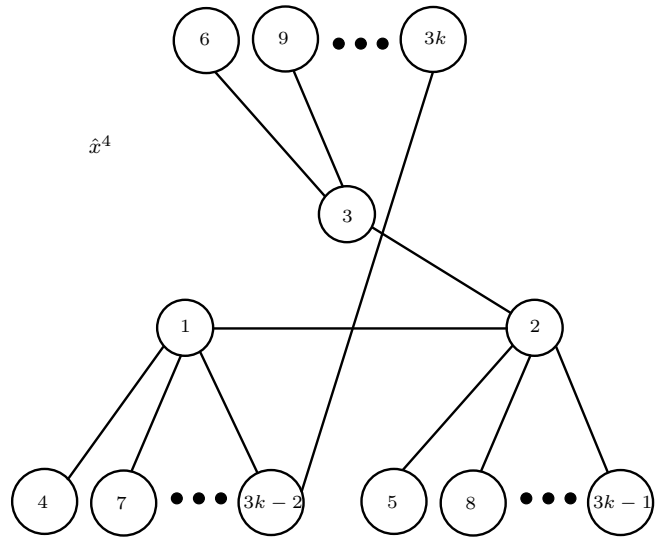

Figura 3.18: Solução viável $\tilde{x}^4$ satisfazendo a desigualdade Blossom na igualdade

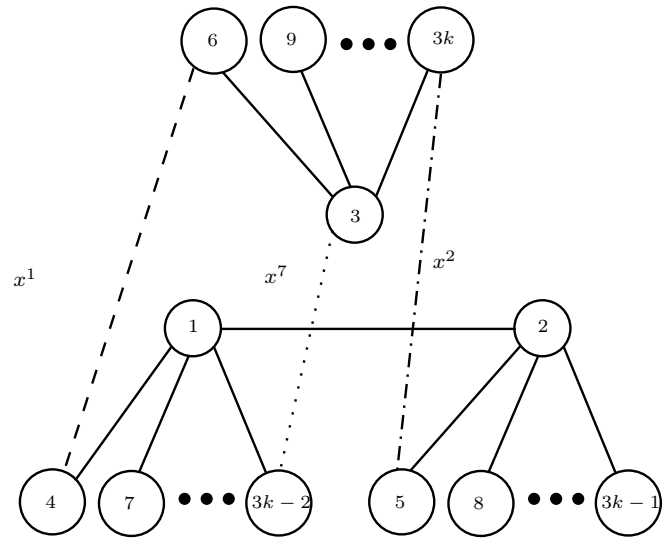

Figura 3.19: Soluções viáveis x^1, x^2, x^7 satisfazendo a desigualdade Blossom na igualdade

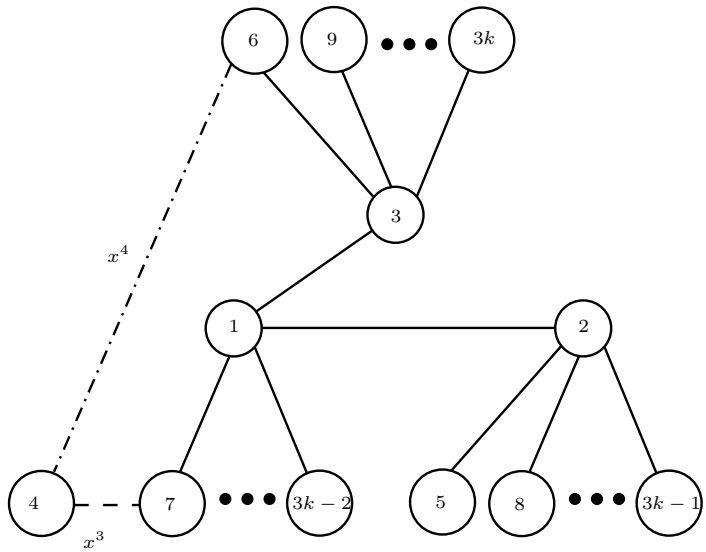

Figura 3.20: Soluções viáveis x^3, x^4 satisfazendo a desigualdade Blossom na igualdade

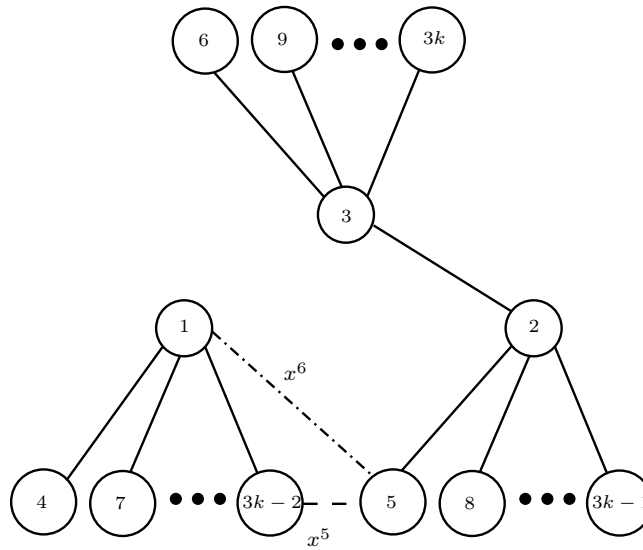

Figura 3.21: Soluções viáveis x^5, x^6 satisfazendo a desigualdade Blossom na igualdade

3.4.3 Dimensões das faces induzidas por desigualdades Combs e Clique Trees

Até o momento, não obtivemos resultados sobre as dimensões das faces induzidas pelas desigualdades Combs e Clique-Trees para o $PARG(n, b), b_i \geq 2,\ \forall i \in V$. Entretanto, podemos estabelecer um caso particular para o qual desigualdades Combs e Clique Trees induzem facetas para $PARG(n, b)$. Para enunciar as condições sob as quais esta propriedade se aplica, considere o seguinte resultado apresentado em Grötschel e Padberg [59].

Lema 3.7 *(Teorema 14 em [59]) Seja $G = (V, E)$ um grafo em $n \geq 6$ vértices. As desigualdades*

1. *Comb*

$$x(E(H)) + \sum_{j=1}^{k} x(E(T_j)) \leq |H| + \sum_{j=1}^{k} (|T_j| - 1) - \frac{k+1}{2},$$

   $k \geq 3$, que satisfazem as propriedades da Definição 3.1 e

2. *Clique-Trees*

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j)) \leq \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{r} (|T_j| - 1) - \frac{s+1}{2},$$

   com pelo menos dois handles, que satisfazem as propriedades da Definição 3.2

definem facetas para $PCV(n)$.

À luz da equivalência poliedral entre $2PARG(n)$ e $PCV(n+1)$ provada por Queyranne e Wang [112], a projeção de desigualdades Combs e Clique Trees que induzem facetas para $PCV(n+1)$ no subespaço $\mathbb{R}^{|E|}$ também induz faceta para $2PARG(n)$. Como o lado direito das desigualdades Combs e Clique-Trees independe do limite de grau dos vértices que encontram-se fora dos handles, os $\frac{n(n-1)}{2} - 1$ caminhos hamiltonianos afim independentes que satisfazem as desigualdades Combs (e Clique-Trees) na igualdade também satisfazem as respectivas desigualdades na igualdade para $PARG(n,b)$, se $b_i = 2, \forall i \in H$ e $b_i \geq 2, \forall i \in V \setminus H$.

3.5 Comentários e perspectivas

Neste Capítulo estudamos o poliedro associado ao PARG e mostramos que algumas desigualdades originalmente propostas para politopos correlatos são também válidas para o PARG. Em particular, mostramos que desigualdades Blossom [35], Combs [58] e Clique-Trees [60] são também válidas para o problema.

O estudo poliedral apresentado para o k-PARG mostrou que as desigualdades da formulação clássica para o problema induzem facetas do respectivo politopo. Além disto, mostramos que um caso particular das desigualdades Blossom (quando $|V| = 3k$ e $|H| = 3$) induzem facetas para o politopo do k-PARG. Esperamos poder obter um resultado de *node-lifting* que permita estender a propriedade de definição de facetas das desigualdades Blossom para casos nos quais $|V| \geq 3k$ e os handles possuam mais de 3 vértices (com $|V|$ *suficientemente grande*).

No decorrer do trabalho, passamos a empregar a seguinte técnica para tentar obter novas desigualdades válidas para o PARG. Como conseguimos otimizar sobre o poliedro $P_2(n,b)$ (isto é, dispomos de algoritmos de separação polinomiais para este poliedro), se obtemos um ponto extremo (fracionário) $\overline{x} \in P_2(n,b)$, analisamos cortes de Gomory associados a $\overline{x}$. A análise de um corte de Gomory com coeficientes inteiros pode enventualmente permitir a caracterização analítica de famílias de desigualdades novas. Através deste processo, obtivemos uma desigualdade que, na forma normalizada, possui coeficientes $1, 2, 3$. Note que nenhuma das desigualdades que cortam o poliedro $P_2(n,b)$ discutidas aqui (Combs ou Clique Trees) possuem coeficiente 3. Até o presente momento, ainda não fomos capazes de caracterizá-la analiticamente. Acreditamos que poderemos fazê-lo no futuro.

Capítulo 4

Um algoritmo Relax-and-Cut para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau

Neste capítulo, propomos um algoritmo do tipo *Non Delayed Relax-and-Cut* (NDRC) para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos vértices. Duas contribuições principais motivaram este trabalho. A primeira delas se origina do fato de enxergarmos as árvores que satisfazem às restrições de grau do PARG como os vértices comuns dos politopos da árvore geradora e do b-matching. Em função disto, pela primeira vez em 25 anos de história do PARG, fomos capazes de fortalecer, pela dualização de desigualdades Blossom em um algoritmo NDRC, os limites inferiores dados pela formulação clássica (3.1) do problema. Numa segunda contribuição, propusemos um procedimento para tentar caracterizar um dado ponto extremo do politopo das árvores geradoras (através de desigualdades SEC (3.3) justas e do hiperplano (3.2) que define a cardinalidade de uma árvore geradora). Combinado com algumas desigualdades Blossom dualizadas durante a execução do algoritmo Relax-and-Cut, este conjunto de desigualdades SECs permite transportar nossos limites duais Lagrangeanos diretamente para um algoritmo de Planos de Corte. Isto é feito de forma extremamente conveniente, não exigindo a resolução de qualquer problema de separação. Os resultados computacionais obtidos indicam que os limites inferiores aqui propostos para o PARG dominam ou são competitivos com os limites Lagrangeanos ou de Relaxação Linear encontrados na literatura. Além disto, nossos limites superiores são competitivos com os melhores da literatura.

4.1 Introdução

O PARG é um problema clássico em Otimização Combinatória com uma longa história na literatura. Essa, aparentemente, se inicia com o algoritmo Branch-and-Bound proposto por Narula e Ho [102]. Naquela referência, além de um algoritmo exato, foram sugeridas duas heurísticas construtivas para se obter soluções viáveis para o problema. Variantes destes dois procedimentos foram amplamente usados desde então. Uma das heurísticas propostas em [102] consiste em uma adaptação do algoritmo de Prim [111] para acomodar a existência das restrições de grau. Na outra, partindo-se de uma árvore geradora de custo mínimo (AGM) para o grafo de definição do problema, tenta-se impor, caso sejam violadas, as restrições de grau nos vértices. Isso é feito através de sucessivas trocas de arestas que reduzem os níveis de violação e mantém a topologia de árvore exigida. Em [102], limites inferiores para o PARG são dados por AGMs definidas em cada nó da árvore de enumeração. Posteriormente, Savelsbergh e Volgenant [117] aprimoraram estes limites inferiores através da implementação de testes de fixação de variáveis baseados em trocas de arestas. A seguir, o algoritmo Branch and Bound de [117] introduziu regras de branching distintas e mais eficientes que aquelas usadas em [102].

Gavish [47] foi o primeiro a empregar Relaxação Lagrangeana em um algoritmo de solução para o PARG. Isso foi feito de forma bastante similar àquela utilizada por Held e Karp [67, 68] para tratar o Problema do Caixeiro Viajante. Após o seu uso em [47], Relaxação Lagrangeana se tornou a *técnica de escolha* para resolução do problema (veja [1, 15, 122], por exemplo).

Outra referência a utilizar Relaxação Lagrangeana para a resolução do PARG é Volgenant [122]. Nessa referência, o autor adaptou o teste de troca de arestas proposto em [117] a um algoritmo Branch and Bound, operando sob um método de *ajuste dual* (para aproximar o valor ótimo do Problema Dual Lagrangeano) no nó raiz da árvore de enumeração. Deste então, o emprego destes testes, que são associados ao cálculo dos custos reduzidos de Programação Linear de uma árvore geradora, se tornou comum em algoritmos Lagrangeanos para o PARG. O algoritmo de Volgenant [122] foi testado em instâncias do k-PARG com até 150 vértices e $k \in \{3, 4, 5\}$. Tais instâncias envolvem tanto custos Euclideanos quanto custos aleatórios.

Outro trabalho na mesma linha daquela descrita acima é a heurística Lagrangeana de Andrade et al. [1]. Os autores aprimoraram as heurísticas do tipo Kruskal

disponíveis na literatura através da introdução de um mecanismo de prevenção de inviabilidades, denominado de *look-ahead mechanism*. Tal dispositivo previne a seleção de arestas que levem eventualmente à geração de componentes conexas sem a capacidade necessária para se conectar aos demais vértices do grafo (por ausência de folga na restrição de grau). O algoritmo em [1] conta ainda com o uso de informação dual Lagrangeana para modificar os custos de entrada do problema e, assim, guiar a geração de soluções viáveis. Outro aspecto relevante de [1] é uma escolha conveniente de um pequeno subconjunto de arestas de E sob o qual o algoritmo irá, inicialmente, operar. O uso deste subconjunto, em uma fase preliminar do algoritmo, permitiu a obtenção de bons limites inferiores e superiores para o PARG, em tempos de computação bastante reduzidos. Em uma fase subsequente do algoritmo, tenta-se obter limites duais mais fortes, utilizando-se o conjunto original de arestas e o melhor limite primal obtido na fase anterior.

Outra linha de trabalho bastante explorada para resolver o PARG, pelo menos aproximadamente, se baseia no uso de metaheurísticas. Dentre as contribuições nesta classe, podemos citar os algoritmos evolutivos de Knowles e Corne [74], Zhou e Gen [127] e Craig et al. [25]. Uma contribuição mais recente nesta linha é a Busca em Vizinhança Variável (VNS) proposta por Ribeiro e Souza [114].

Voltando aos métodos exatos de solução, Caccetta e Hill [15] propuseram um algoritmo Branch-and-Cut para resolver o problema. Nessa referência, um algoritmo bastante similar ao procedimento de Relaxação Lagrangeana de Volgenant [122] é inicialmente usado como pre-processamento para o algoritmo exato. Este, por sua vez, é baseado em heurísticas para a separação de cutsets e SECs. Um aspecto que colaborou positivamente para a boa qualidade dos resultados obtidos em [15] foi a técnica empregada para prevenir e identificar a ocorrência de *tailing-off*. Essa técnica é similar àquela empregada por Padberg e Rinaldi [108] em um algoritmo Branch-and-Cut para o PCV. O algoritmo em [15] foi testado para instâncias do k-PARG de até 800 vértices, com custos aleatórios e $k \in \{3, 4, 5, 6\}$.

Neste Capítulo, introduzimos um algoritmo NDRC para o PARG. O algoritmo difere daqueles encontrados na literatura em vários aspectos. Em primeiro lugar, empregamos desigualdades Blossom em um esquema NDRC. Em função disto, via de regra, conseguimos limites duais mais fortes que os disponíveis na literatura, sejam eles Lagrangeanos ou de Programação Linear. Por outro lado, nossa heurística primal para o PARG, embora inspirada na heurística Lagrangeana de [1], dela se

diferencia pela Busca Local aqui empregada e pelo fato de que nos beneficiamos de informação dual Lagrangeana de melhor qualidade, já que trabalhamos com uma formulação mais forte para o problema. Por fim, introduzimos um procedimento de identificação de desigualdades que é válido para problemas que admitem relaxações Lagrangeanas definidas sobre matróides. Esse procedimento, pelo menos nos testes aqui realizados, permitiu transportar para um algoritmo de Planos de Corte, os melhores limites Lagrangeanos obtidos pelo algoritmo NDRC.

Este Capítulo é organizado da seguinte forma. Na Seção 4.2, descrevemos os procedimentos empregados para obtenção de limites Lagrangeanos mais fortes para o PARG. Na seção 4.3, o procedimento de identificação de desigualdades que permite transportar os limites duais Lagrangeanos para um algoritmo de Planos de Corte é detalhado. Os experimentos computacionais que realizamos são descritos na Seção 4.4 e, finalmente, na Seção 4.5, encerramos o capítulo com alguns comentários.

4.2 Limites Lagrangeanos para o PARG

Na formulação (3.1) do PARG, assuma que multiplicadores $\lambda \in \mathbb{R}_+^{|V|}$ sejam empregados para dualizar, de forma Lagrangeana, as desigualdades (3.4). Limites inferiores válidos para o problema são então fornecidos pelo Problema Lagrangeano PL(λ) definido como:

$$z_\lambda = \min \left\{ \sum_{e \in E} c_e x_e + \sum_{i \in V} \lambda_i (x(\delta(i)) - b_i) : x \text{ é uma árvore geradora de } G \right\}. \tag{4.1}$$

O melhor limite inferior possível de se obter a partir de PL(λ) é dado pelo Problema Dual Lagrangeano (PDL)

$$z_d = \max \{ z_\lambda : \lambda \geq 0 \}. \tag{4.2}$$

Uma decomposição alternativa para a formulação (3.1) é aquela em que cada Problema Lagrangeano consiste em determinar um b-matching 1-capacitado de mínimo custo Lagrangeano. Uma vez que os vértices dos politopos $CM(n, b, \mathbb{1})$ e $AGM(n)$ são inteiros, isto é, satisfazem à Propriedade de Integralidade (veja o Capítulo 2), os melhores limites dados por (4.2) e pelo dual Lagrangeano associado a esta relaxação alternativa serão iguais. Em princípio, uma desvantagem da relaxação alternativa seria, provavelmente, um tempo de computação mais elevado para se resolver cada PL(λ). A observação é certamente pertinente pois a resolução de problemas de

b-matching com pesos nas arestas custa, no pior caso, $O(n^3)$ (veja, por exemplo Gerards [50]), enquanto a determinação de árvores geradoras de custo mínimo pode ser obtida em tempo limitado por $O(m\log(n))$ (veja [104]). Assim sendo, não nos parece vantajosa a implementação de um algoritmo Relax-and-Cut baseado nesta relaxação alternativa.

O algoritmo NDRC implementado aqui baseia-se na dualização dinâmica de desigualdades Blossom para tentar obter limites inferiores mais fortes que (4.2). Para a obtenção (aproximação) destes limites empregamos o Método do Subgradiente (MS) adaptado para algoritmos NDRC, conforme detalhado no Capítulo 2. Um aspecto muito importante neste procedimento é a identificação de desigualdades Blossom violadas pelas soluções de cada um dos PL(λ) formulados ao longo da aplicação do MS. Este aspecto é tratado em detalhes na Seção a seguir.

4.2.1 Separação de desigualdades Blossom em um algoritmo NDRC

Em linhas gerais, um algoritmo NDRC onde desigualdades Blossom são dualizadas dinamicamente funciona da seguinte forma. Para toda iteração do MS (com exceção da primeira), PL(λ) pode conter, além das desigualdades (3.4), desigualdades Blossom dualizadas no decorrer da execução do MS. Diferentemente das desigualdades (3.4), uma desigualdade Blossom é dualizada *on-the-fly*, ou seja, é dualizada apenas quando se torna violada pela solução do PL(λ) corrente. Para facilitar a exposição, a partir de agora, particionamos λ em (τ, γ), onde $\tau \in \mathbb{R}_+^{|V|}$ são os multiplicadores associados às desigualdades (3.4) e γ (cuja dimensão pode variar ao longo das iterações do MS) são os multiplicadores associados às desigualdades Blossom que se encontram dualizadas.

Para melhor compreender o algoritmo proposto para a separação de desigualdades Blossom, denote por $\overline{x}^k$ a solução associada a z_{λ^k}, obtida ao resolver o Problema Lagrangeano na iteração k do MS. Para a separação desta classe de desigualdades, os algoritmos disponíveis na literatura (veja o Apêndice B) assumem que as desigualdades de grau (3.4) não são violadas no ponto $\overline{x}^k$ investigado. No entanto, para nossa aplicação específica, isso em geral não ocorre. De qualquer modo, o fato de $\overline{x}^k$ definir uma árvore geradora de G pode ser utilizado a nosso favor. Baseado neste fato, desenvolvemos uma heurítica gulosa simples para efetuar a separação. O procedimento é formalmente descrito no Algoritmo 4.1, que se segue. Nesse algoritmo,

dado que $\overline{x}^k$ corresponde a uma árvore geradora de G, descrevemos o procedimento de separação em termos dos vértices e arestas dessa árvore. A heurística gulosa (passos 2 a 28) é executada uma vez para cada vértice associado a uma desigualdade (3.4) violada por $\overline{x}^k$. Em cada execução, a heurística é inicializada com um conjunto H contendo um único vértice (passo 4). A seguir, tentamos expandir o conjunto H inicial. Este processo de expansão persiste enquanto a introdução de novos vértices induzir uma desigualdade Blossom violada. Os vértices candidatos a expandir H são os vizinhos dos vértices daquele conjunto, na árvore induzida por $\overline{x}^k$ (passo 7). Assim, para cada vértice candidato a entrar no conjunto H no passo corrente, avaliamos o conjunto de arestas T associado à sua inserção (passos 9 a 13) e a violação da desigualdade Blossom daí decorrente (passo 14). Note que o conjunto de arestas T associado à inserção de um vértice candidato é um subconjunto de máxima cardinalidade de $(\delta(H) \cap \{e \in E : \overline{x}_e^k = 1\})$, com $b(H) + |T|$ ímpar. Após analisarmos separadamente o impacto da inserção de cada um dos vértices vizinhos de H, a expansão desse conjunto se dá pela introdução do vértice que induz a desigualdade mais violada localmente (critério guloso indicado pelo passo 16). Uma vez finalizado o processo de expansão, dualizamos a desigualdade que induziu a maior violação durante todo o processo de expansão. Note que o procedimento só dualiza, eventualmente, desigualdades que não se encontrem explicitamente dualizadas.

Como exemplo da aplicação do Algoritmo 4.1, assuma que $\overline{x}^k$ induz a árvore indicada na Figura 4.1. Assuma também que os dois círculos preenchidos em negro (vértices) naquela Figura sejam associados às desigualdades (3.4) violadas. Os vértices delimitados por cada camada sucessiva de linhas intermitentes indicam possíveis conjuntos sucessivos de vértices H_1, H_2 and H_3, gerados a partir da expansão do conjunto H inicializado com o vértice escuro mais à esquerda. Deve ser observado que dois conjuntos sucessivos H_i e H_{i+1} diferem apenas por um vértice (que deve pertencer ao conjunto de vértices vizinhos a H_i).

4.2.2 Limites superiores

Um outro aspecto importante para um bom desempenho de um algoritmo NDRC é a disponibilidade de procedimentos que permitam obter soluções viáveis de boa qualidade para o problema tratado. Para o PARG, utilizamos informação dual Lagrangeana em uma heurística gulosa do tipo Kruskal [77]. A motivação fundamental que permeia esta abordagem é a idéia intuitiva (validada por algoritmos primais-

Algoritmo 4.1 Algoritmo de Separação de Desigualdades Blossom para NDRC

1: $\overline{V} := \{i \in V : \ \overline{x}^k(\delta(i)) > b_i\}$
2: **para todo** $i \in \overline{V}$ **faça**
3: $maxviol \leftarrow -\infty$
4: $H \leftarrow \{i\}$
5: $viol = \overline{x}^k(\delta(i)) - b_i$
6: **enquanto** $(viol > 0)$ **faça**
7: $N(H) = \{j \in V \ : \ \overline{x}^k_{(i,j)} = 1, \text{para algum } \ i \in H\}$
8: **para todo** $j \in N(H)$ **faça**
9: **Se** $b(H \cup \{j\}) + |N(H \cup \{j\})|$ é ímpar **então**
10: $T(j) \leftarrow \delta(H \cup \{j\})$
11: **senão**
12: $T(j) \leftarrow \delta(H \cup \{j\}) \setminus \{e'\}$, onde e' é qualquer aresta em $\delta(H \cup \{j\})$
13: **fim Se**
14: $viol(j) = \frac{b(H)+|T(j)|-1}{2} - \overline{x}^k(E(H \cup \{j\}) - \overline{x}^k(T(j))$
15: **fim para**
16: Determine $z \ : \ z \in \text{argmax}_{j \in N(H)}\{viol(j)\}$
17: $viol = viol(z)$
18: $H \leftarrow H \cup \{z\}$
19: $T \leftarrow T(z)$
20: **Se** $(viol > maxviol)$ **então**
21: $BESTH \leftarrow H$
22: $BESTT \leftarrow T$
23: $maxviol \leftarrow viol$
24: **fim Se**
25: **fim enquanto**
26: **Se** $(maxviol > 0)$ e (a desigualdade gerada por $HBEST, TBEST$ não foi dualizada) **então**
27: dualize-a
28: **fim Se**
29: **fim para**

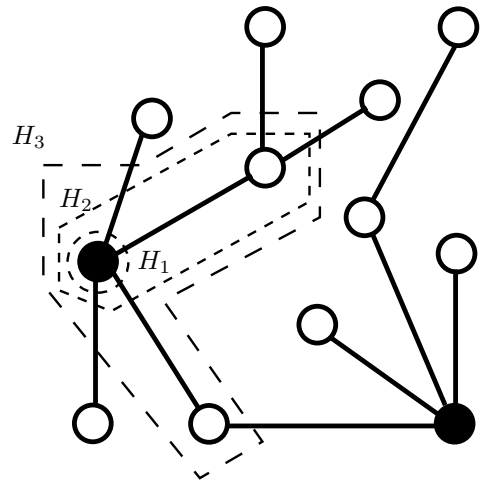

Figura 4.1: Exemplo da aplicação do algoritmo de separação de BIs.

duais) de que as soluções duais (Lagrangena ou de Programação Linear) carregam informação relevante para gerar soluções primais de boa qualidade.

Operando sob Relaxação Lagrangeana, nosso algoritmo para obtenção de limites primais envolve dois estágios principais. O primeiro é um algoritmo construtivo onde, a exemplo do procedimento de Kruskal, arestas de E são selecionadas para expandir uma floresta $\mathcal{F} = (V, E_{\mathcal{F}})$ (inicialmente contendo vértices isolados), uma de cada vez, em uma ordem não decrescente de custos. Considere então uma aresta $e = (i, j)$ candidata a entrar em $\mathcal{F}$. Essa aresta só será eventualmente aceita para inserção se forem satisfeitas as três condições a seguir:

- $E_{\mathcal{F}} \cup \{(i, j)\}$ não possui ciclos;
- $|\delta(i) \cap E_{\mathcal{F}}| \leq b_i - 1$ e $|\delta(j) \cap E_{\mathcal{F}}| \leq b_j - 1$;
- se $|E_{\mathcal{F}}| \neq |V| - 2$, então $|\delta(i) \cap E_{\mathcal{F}}| + |\delta(j) \cap E_{\mathcal{F}}| > 2$.

A segunda das três condições acima procura garantir, na iteração corrente do algoritmo, a viabilidade da solução em construção. A terceira condição (o mecanismo look-ahead [1]) vai um passo além. Ela tenta preservar a capacidade de conexão daquela componente conexa de $\mathcal{F}$ que conteria (i, j) (caso essa aresta fosse agora introduzida), nas iterações futuras do procedimento.

A fase construtiva do algoritmo é chamada a cada iteração do MS, diante de *custos complementares* $\{(1 - \overline{x}_e^k)c_e : e \in E\}$ onde, como antes, $\overline{x}^k$ é a solução de PL(λ^k). Deve ser destacado que, com o uso de custos complementares, induzimos a

heurística a selecionar o maior número possível de arestas presentes na solução dual $\overline{x}^k$.

A fase seguinte do algoritmo se constitui em uma Busca Local (BL). Nesta fase, voltamos a utilizar os custos originais $\{c_e : e \in E\}$ em detrimento dos custos complementares. A idéia central da BL é investigar soluções vizinhas à árvore viável $\mathcal{T} = (V, E_{\mathcal{T}})$ obtida ao final da fase construtiva do algoritmo. Isso é feito na expectativa de obter soluções mais baratas que aquela representada por $\mathcal{T}$. Especificamente, investigamos a *1-vizinhança* de $\mathcal{T}$, isto é, o conjunto das árvores geradoras de G que satisfazem às restrições de grau e diferem de $\mathcal{T}$ por exatamente uma aresta. Sempre que uma árvore viável mais barata que $\mathcal{T}$ é encontrada nessa vizinhança, $\mathcal{T}$ é substituída pela nova árvore e o procedimento BL é repetido. Caso contrário, BL é encerrado.

Ao implementar BL, nos defrontamos com as alternativas de aceitar a primeira solução viável encontrada cujo custo é inferior ao custo da árvore corrente $\mathcal{T}$ (estratégia *first improvement*) ou, caso contrário, escolher a solução mais barata de toda a 1-vizinhança de $\mathcal{T}$ (estratégia *best improvement*), se esta tiver uma solução inferior a $\mathcal{T}$. Após a realização de testes computacionais, optamos pela estratégia *first improvement*, por produzir, em menor tempo de computação, soluções tão boas quanto aquelas obtidas pela estratégia *best improvement*.

Com o intuito de permitir o uso da heurística descrita acima à cada iteração do MS, sem provocar, em contra-partida, um crescimento excessivo do tempo de processamento, implementamos alguns testes de dominância, muitos dos quais foram originalmente propostos em [113, 114]. Estes testes se mostravam bastante eficazes. Em uma etapa inicial de BL, dada a solução inicial $\mathcal{T}$, calculamos os seguintes parâmetros:

- $p(r, i)$: o custo da aresta mais cara do caminho (único) em $\mathcal{T}$ entre i e um vértice raiz qualquer r, para todo $i \in V, i \neq r$. Por definição, $p(r, r) = 0$;
- $a(i) = \max\{c_e : e \in \delta(i) \cap E_{\mathcal{T}}\}, \forall i \in V$, ou seja, o custo da aresta mais cara incidente a i em $\mathcal{T}$;
- c^*: o custo da aresta mais cara em $\mathcal{T}$.

Em função da vizinhança escolhida, qualquer aresta candidata a reduzir o custo de $\mathcal{T}$ deve possuir custo inferior a c^*. Assim, todas as arestas que não satisfizerem

esta condição podem ser descartadas, de antemão. Suponha agora que estejamos investigando a inserção de uma aresta candidata $e = (i, j)$ em uma árvore viável $\mathcal{T}$, corrente. Se as extremidades da aresta não são saturadas, isto é, se $|\delta(i) \cap E_{\mathcal{T}}| \leq b_i - 1$ e $|\delta(j) \cap E_{\mathcal{T}}| \leq b_j - 1$, a eliminação de qualquer aresta no caminho entre i e j em $\mathcal{T}$ produz uma árvore viável. Contudo, antes de determinar a aresta de máximo custo neste caminho, verificamos se $c_{(i,j)} < p(i, r)$ ou $c_{(i,j)} < p(j, r)$. É evidente que se ambas as condições não forem satisfeitas, o custo da aresta mais cara no caminho entre i e j em $\mathcal{T}$ não pode superar $c_{(i,j)}$.

De forma a aumentar a eficiência destes testes, não escolhemos a raiz r como um vértice qualquer de V. Idealmente, r deve ser um vértice que tenha grau maior que 1 em $\mathcal{T}$ e cuja distância a uma folha de $\mathcal{T}$ é máxima. Determinar (e atualizar) tal vértice pode ser computacionalmente custoso. Assim, optamos por escolher r aleatoriamente dentre os vértices de máximo grau em $\mathcal{T}$.

Para encerrar a descrição dos testes de dominância implementados, suponha que uma das extremidades de uma aresta $e = (i, j)$, digamos j, esteja, ao contrário de i, saturada, ou seja: $|\delta(j) \cap E_{\mathcal{T}}| = b_j$ e $|\delta(i) \cap E_{\mathcal{T}}| \leq b_i - 1$. Note que, nesse caso, (i, j) só será uma opção de melhora quando $c_{(i,j)} < a(j)$.

Em nossa implementação, sempre que uma árvore $\mathcal{T}$ é modificada, todos os parâmetros envolvidos nos testes de dominância são recalculados eficientemente através das estruturas de dados propostas em [52].

A diferença principal entre a heurística primal proposta aqui e aquela proposta em [1] reside nos refinamentos discutidos acima. No entanto, vale também ressaltar que dispomos aqui de informação dual de muito melhor qualidade que aquela disponível em [1]. Essa informação é proveniente dos limites inferiores gerados pelo algoritmo NDRC que, devido à dualização de desigualdades Blossom, dominam os limites inferiores gerados em [1]. Como conseqüência, nossos limites superiores são certamente competitivos com os de [1]. Um outro aspecto que muito provavelmente contribuiu de forma positiva para a qualidade de nossos limites superiores e inferiores são os testes de pre-processamento e fixação de variáveis, descritos na próxima Seção.

4.2.3 Pré-processamento, testes de fixação de variáveis e ciclos de MS

Uma regra simples a se aplicar a uma instância do PARG, antes de tentar resolvê-la, é a seguinte. Sempre que $|V| > 2$ e $b_i = b_j = 1$, para qualquer $i, j \in V$, a aresta $e = (i, j)$ pode ser eliminada de E, já que não existe para o problema qualquer solução viável que a contenha.

Além do teste simples de pré-processamento descrito acima, utilizamos também testes de fixação de variáveis. Em uma iteração k de MS, a solução $\overline{x}$ de PL(λ^k) e seu valor correspondente z_{λ^k} podem ser combinados com limites superiores válidos para o problema, numa tentativa de fixar arestas *dentro* ou *fora* de qualquer solução ótima. Isto é realizado através do uso de *custos reduzidos* de Programação Linear (CRPL). Para o caso específico de árvores geradoras, CRPLs podem ser obtidos através da troca conveniente de arestas da árvore geradora de G, induzida por $\overline{x}$ [52]. Para entender como esses testes são conduzidos, considere uma AGM de G. Claramente, esta árvore fornece um limite inferior válido para o custo da solução ótima de um PARG definido em G. O CRPL associado a uma aresta $e \in E$ (variável x_e), em relação à AGM, é dado pela diferenca entre o custo de e e o custo da aresta mais cara no (único) caminho entre as extremidades de e ao longo da árvore. Sempre que uma aresta $e \in E$ possuir custo reduzido maior que a diferença entre um limite superior e o correspondente limite inferior associado à AGM, esta aresta é garantidamente sub-ótima. No decorrer do processo de fixação de variáveis por custos reduzidos, se todas as arestas incidentes a um vértice i, com exceção de uma, tiverem sido fixadas *fora* de qualquer solução ótima, o vértice i e sua única aresta incidente são removidos do grafo (já que aquela aresta é necessariamente ótima).

Após aplicar MS a uma Relaxação Lagrangeana do PARG, freqüentemente a otimalidade do melhor limite superior disponível pode não ter sido comprovada. Neste caso, se os limites inferiores e superiores forem suficientemente fortes e se um *número significativo* de variáveis tiverem sido eliminadas através dos testes de fixação de variáveis descritos acima, aplicamos novas rodadas de MS, sucessivamente, como sugerido em [87, 88]. Isto é feito enquanto a otimalidade do melhor limite superior disponível não tiver sido comprovada e enquanto um número significativo de arestas de G forem eliminadas na rodada corrente do MS.

A idéia de preprocessar instâncias de um problema previamente à aplicação de um algoritmo exato tem sido usada com freqüência na literatura. Exemplos podem

ser encontrados, entre outros, em [33] (testes baseados em propriedades específicas do problema considerado), em [15] (testes baseados em custos reduzidos como descritos acima) e em [16] (testes baseados em propriedades estruturais e nos CRPLs do problema tratado). Embora não seja propriamente uma técnica de pré-processamento, a metodologia a ser descrita na próxima Seção permite acelerar a resolução exata do PARG (bem como de algumas outras classes de problemas de Otimização Combinatória) por um algoritmo Branch-and-Cut.

4.3 Atalhos Lagrangeanos para algoritmos de Planos de Corte

Nos algoritmos Relax-and-Cut, além da dualização de desigualdades como tradicionalmente feito em algoritmos de Relaxação Lagrangeana Clássica (isto é, de uma forma estática, e com um pequeno número de desigualdades sendo dualizadas), desigualdades são também dualizadas dinamicamente (isto é, são incorporadas ou eliminadas das relaxações, de acordo com algum critério de melhora dos limites duais). Esta característica permite que algoritmos Relax-and-Cut sejam capazes de lidar adequadamente com formulações que contém um número exponencial de desigualdades (na medida em que se dualiza apenas uma pequena fração delas). Dessa forma, algoritmos Relax-and-Cut permitem que limites inferiores de Relaxação Lagrangeana sejam diretamente transportados para um algoritmo de Planos de Corte sem a necessidade de recorrer a algum algoritmo de separação. Um exemplo disso pode ser encontrado em [16], onde um algoritmo NDRC, baseado na dualização dinâmica de desigualdades Clique, foi proposto para o Problema de Particionamento de Conjuntos com Restrição de Cardinalidade.

Nesta Seção discutimos uma forma de acelerar um algoritmo de Planos de Corte aplicado a um contexto mais complexo que aquele tratado em [16]. A técnica aqui discutida, como veremos a seguir, pode ser estendida a toda uma classe de Problemas de Otimização Combinatória.

Considere então um algoritmo exato híbrido onde NDRC é usado como pré-processamento para um algoritmo Branch-and-Cut. Neste algoritmo, NDRC é utilizado para tentar fixar variáveis *dentro* ou *fora* de soluções ótimas (através do uso combinado da heurística Lagrangeana descrita na Subseção 4.2.2, limites inferiores e testes de fixação de variáveis apresentados na Subseção 4.2.3). Um algoritmo

desta natureza é similar àquele proposto em [15] para o PARG (um procedimento de ajuste dual é usado em [15] ao invés do algoritmo NDRC sugerido aqui). Contudo, em nosso algoritmo híbrido, procuramos também transportar para o algoritmo de Planos de Corte, em um *warm-start*, os (idealmente fortes) limites inferiores obtidos pelo algoritmo NDRC. Para conseguir isto, considere z_{λ^*}, o melhor limite dual Lagrangeano encontrado ao longo da aplicação de MS, seu PL(λ^*) correspondente e a solução ótima $\overline{x}^*$ para PL(λ^*). Considere também as desigualdades Blossom que foram explicitamente dualizadas (em conjunto com as desigualdades (3.4), na iteração do MS na qual z_{λ^*} foi obtido). Claramente, todas estas desigualdades Blossom deveriam ser incluídas na Relaxação Linear que pretendemos construir. Adicionalmente, seria também desejável identificar e incluir nessa relaxação um número mínimo de desigualdades SECs necessárias (em conjunto com desigualdades (3.2) e (3.5)) para caracterizar a árvore geradora de G induzida por $\overline{x}^*$ (ou seja, gostaríamos de indentificar o menor conjunto de hiperplanos associados às desigualdades (3.2), (3.3) e (3.5), cuja interseção resulta em $\overline{x}^*$). Se uma Relaxação Linear do PARG incluísse todas as desigualdades (3.2), (3.4) e (3.5) e ainda as Blossom e SECs mencionadas acima, o limite dual a ela associado seria, garantidamente, tão forte quanto z_{λ^*}. Considerando que ainda não dispomos de uma caracterização completa das SECs necessárias a isso, propusemos uma estratégia que, para os experimentos computacionais desenvolvidos neste trabalho, produziu limites inferiores de Relaxação Linear praticamente tão fortes (na maioria das vezes mais fortes) que z_{λ^*}. Esta estratégia é descrita a seguir.

Assuma que, na iteração de MS onde z_{λ^*} foi obtido, as arestas em E sejam ordenadas em magnitude crescente de seus custos Lagrangeanos (empates são resolvidos aleatoriamente). Uma AGM de G, diante destes custos, pode ser gerada, por exemplo, através do algoritmo de Kruskal [77]. Neste caso, as arestas são inseridas na árvore em construção (se não induzem ciclos com as arestas previamente escolhidas), uma de cada vez. Após $|V|-1$ inclusões de arestas, uma AGM de G é obtida. Assuma que esta AGM é formada (na ordem em que as arestas foram selecionadas pelo algoritmo de Kruskal) pelas arestas $\{e_1, \ldots, e_{|V|-1}\}$. Claramente, cada uma destas arestas, quando incluída na árvore em construção, origina uma componente conexa (eventualmente contendo apenas dois vértices). Denotamos por *conjuntos gulosos* $S_j \subseteq V$ os conjuntos de vértices da componente conexa induzida pela inserção da aresta e_j, para $j = 1, \ldots, |V|-1$.

Motivados por uma prova da otimalidade do algoritmo de Kruskal proposta em [94], transcrita a seguir, incorporamos em nossa Relaxação Linear inicial do PARG aquelas SECs associadas respectivamente aos conjuntos gulosos $\{S_j : \ j = 1, \ldots, |V| - 2\}$.

Para compreender o argumento teórico que motivou esta estratégia, considere por um instante o problema de determinar uma AGM diante de custos não negativos $\{c_e : e \in E\}$. Matematicamente, o problema pode ser definido como

$$\min \left\{ \sum_{e \in E} c_e x_e : x \in \mathbb{R}^{|E|} \text{ , sujeito a } (3.2), (3.3), (3.5) \right\} \tag{4.3}$$

e seu dual pode ser escrito como:

$$\max -\mu_V (n-1) - \sum_{\emptyset \neq S \neq V} \mu_S (|S| - 1) \tag{4.4}$$

sujeito a:

$$-\mu_V - \sum_{S : e \in E(S)} \mu_S \leq c_e, \ \forall e \in E, \tag{4.5}$$

$$\mu_s \geq 0, \ \forall S \neq V, S \neq \emptyset. \tag{4.6}$$

Suponha agora que o algoritmo de Kruskal foi aplicado ao problema (4.3) e que as arestas (ordenadas em magnitude não decrescente de seus custos) associadas à árvore geradora ótima de G obtida sejam $\{f_1, f_2, ..., f_{|V|-1}\}$. Recorde que definimos S_i como o subconjunto de vértices de V na componente conexa induzida pela introdução da aresta f_i pelo algoritmo de Kruskal. Assuma agora que valores para as variáveis duais definidas acima sejam dados por:

- $\mu_V = -c_{f_{|V|-1}}$;
- $\mu_{S_i} = c_{f_k} - c_{f_i}$, $i = 1, \ldots, |V| - 2$, onde $k > i$ e f_k é a aresta empregada pelo método de Kruskal para conectar a componente conexa formada pelo conjunto de vértices S_i a alguma outra componente conexa, em uma iteração k posterior do algoritmo. Já que $i = 1, ..., |V| - 2$, sempre há uma aresta f_k adequadamente definida para cada conjunto S_i. Note ainda que $c_{f_k} \geq c_{f_i}$.
- $\mu_S = 0$, para qualquer outro subconjunto S de V.

Note que esses valores são viáveis (cada μ_S é não negativo e μ_V é irrestrita em sinal). Considere agora o vetor de incidência associado à árvore obtida pelo

algoritmo de Kruskal, isto é, o vetor $\overline{x}$ tal que $\overline{x}_e = 1$ se $e \in \{f_1, f_2, ..., f_{|V|-1}\}$ e $\overline{x}_e = 0$, caso contrário. Note também que $\{\mu_S : \emptyset \neq S \subset V\}, \mu_V$ e o vetor de incidência $\overline{x}$ satisfazem às condições de complementaridade-folga para o par de problemas (4.4)-(4.6) e (4.3), uma vez que $-\mu_V - \sum_{S:f_i \in E(S)} \mu_S = c_{f_i}$.

Vamos agora considerar a contribuição dos custos c_{f_i} para o valor da função objetivo do programa dual (4.4). Cada uma das arestas f_i contribui em dois termos desta função, a saber:

- no termo referente à componente S_i, isto é, $-\mu_{S_i}(|S_i|-1) = -(c_{f_k} - c_{f_i})(|S_i|-1)$, onde f_k é a aresta anteriormente definida se $i \leq |V|-2$. Para a aresta $f_{|V|-1}$, esta contribuição é $c_{f_{|V|-1}}(|V|-1)$.

- nos termos referentes às componentes conexas P_i e Q_i que foram unidas pela introdução da aresta f_i, isto é: $-(c_{f_i} - c_{f_{j_1}})(|P_i|-1)$ e $-(c_{f_i} - c_{f_{j_2}})(|Q_i|-1)$, $j_1 \neq j_2$, $j_1 < i$ e $j_2 < i$. Note que $P_i = S_{j_1}$ e $Q_i = S_{j_2}$, para $j_1 \neq j_2$, $j_1 < i$ e $j_2 < i$.

Diante das observações acima, (4.4) pode ser reescrita como:

$$\begin{aligned} -\mu_V(n-1) - \sum_{S \neq V} \mu_S(|S|-1) &= -\mu_V(n-1) - \sum_{i=1}^{|V|-2} \mu_{S_i}(|S_i|-1) = \\ &= \sum_{i=1}^{|V|-1} c_{f_i}[|S_i| - 1 - (|P_i|-1) - (|Q_i|-1)] = \\ &= \sum_{i=1}^{|V|-1} c_{f_i}, \end{aligned} \tag{4.7}$$

já que $|S_i| = |Q_i| + |P_i|$ e $|V| = |Q_{|V|-2}| + |P_{|V|-2}|$.

Como o valor da solução obtida pelo algoritmo de Kruskal é igual a (4.7), temos uma prova da otimalidade deste algoritmo. Este resultado permite dizer que as desigualdades SECs associadas aos conjuntos gulosos são suficientes para se provar a otimalidade da solução obtida pelo algoritmo de Kruskal. Note que dentre as desigualdades SEC satisfeitas na igualdade por $\{f_1, f_2, ..., f_{|V|-1}\}$ apenas aquelas definidas pelos conjuntos gulosos possuem multiplicadores positivos e portanto colaboram com o valor de (4.4). Todas as demais desigualdades SEC satisfeitas na igualdade por $\{f_1, f_2, ..., f_{|V|-1}\}$ possuem variáveis duais associadas nulas.

Por outro lado, se associarmos multiplicadores $\mu_V \in \mathbb{R}$ e $\{\mu_S \in \mathbb{R}_+ : \emptyset \neq S \subset V\}$ às desigualdades (3.2) e (3.3), respectivamente, um limite inferior válido para (4.3)

é dado por

$$\min \quad \left\{ \sum_{e \in E} c_e x_e \quad + \right.$$

$$\mu_S(\textstyle\sum_{e \in E} x_e - (n-1)) \quad +$$

$$\left. \sum_{\emptyset \neq S \subset V} \mu_S \left[\sum_{e \in E(S)} x_e - (|S|-1) \right] : \ 0 \leq x_e \leq 1, e \in E \right\},$$

que pode ser reescrito como

$$\min \left\{ \sum_{e \in E} c'_e x_e - \mu_S(n-1) - \sum_{\emptyset \neq S \subset V} \mu_S(|S|-1) : 0 \leq x_e \leq 1, e \in E \right\}, \qquad (4.8)$$

onde $c'_e := c_e + \sum_{S: e \in E(S)} \mu_S + \mu_V$ são custos reduzidos associados às arestas $e \in E$. Está claro que os multiplicadores $\{\mu_S \in \mathbb{R}_+ : \emptyset \neq S \subset V\}$ e μ_V definidos anteriormente são válidos para (4.8) e que o limite inferior dado por (4.8) também iguala o custo da solução de Kruskal. Da mesma maneira, apenas os Multiplicadores de Lagrange associados às SECs geradas pelos conjuntos gulosos são necessárias para a prova de otimalidade do algoritmo de Kruskal.

Retornemos agora à questão de caracterizar um ponto extremo do politopo $AGM(n)$ das árvores geradoras. Para este propósito, são necessárias $\frac{n(n-1)}{2} - 1$ desigualdades linearmente independentes, satisfeitas na igualdade por $\{f_1, f_2, ..., f_{|V|-1}\}$. Embora as SECs associadas aos conjuntos gulosos não sejam suficientes para esta caracterização, elas carregam informação suficiente para a determinação do valor da função objetivo ótima, no caso da AGM induzida por $\{f_1, f_2, ..., f_{|V|-1}\}$. Assim sendo, nos pareceu razoável, na ausência de uma caracterização completa, utilizar as desigualdades SECs geradas a partir dos conjuntos gulosos em conjunto com as desigualdades Blossom explicitamente dualizadas em PL(λ^*), para obter um warm-start para um algoritmo de Planos de Corte. Nossa expectativa ao fazer isso era de que o valor obtido para essa Relaxação Linear inicial do PARG fosse próxima a z_{λ^*}.

A idéia apresentada acima pode ser estendida, de forma direta, a outros problemas que possuem Relaxações Lagrangeanas definidas em matróides. Isto se aplica porque a prova de otimalidade do algoritmo de Kruskal, apresentada nesta Seção, é

um caso particular da prova de otimalidade de algoritmos gulosos para a solução de problemas de otimização definidos em matróides (veja também a Proposição 3.3, p. 669, em Nemhauser e Wolsey [104], para maiores detalhes). Pelo que conhecemos, esta é a primeira vez que conjuntos gulosos são empregados na tentativa de identificar um conjunto de desigualdades válidas *atraentes*, para serem transportadas de uma Relaxação Lagrangeana para algoritmos de Planos de Corte.

Desta forma, o procedimento de warm-start proposto aqui é mais elaborado que aquele proposto em [16]. Isto se aplica porque a justificativa para sugerir *boas* desigualdades Blossom para um algoritmo de Planos de Corte depende do problema sendo tratado (como é o caso também das desigualades Clique [103] empregadas em [16]). Contudo, a justificativa para levar *boas* desigualdades SECs (de um algoritmo NDRC para um algoritmo de Planos de Corte) não o é.

Devemos salientar também que o nosso procedimento de warm-start é independente dos testes de fixação de variáveis (embora obviamente se beneficie deles). Portanto, nosso warm-start se diferencia bastante da abordagem sugerida em [15], que também pode ser considerada, indiretamente, um warm-start para um algoritmo de Planos de Corte (através das eventuais reduções que um pré-processamento baseado em Relaxação Lagrangeana pode provocar nas dimensões das instâncias).

Na Seção seguinte, mostramos, dentre outros resultados, que o procedimento aqui proposto levou, pelo menos para o conjunto de instâncias testadas, a limites inferiores iniciais muito fortes para o PARG.

4.4 Experimentos computacionais

Nesta Seção apresentamos os resultados computacionais obtidos ao empregar o algoritmo NDRC, proposto neste Capítulo, para o PARG. O primeiro conjunto de instâncias testado, denominado MGRAPH, foi introduzido por Craig et al. [25] e tem $|V|$ variando entre 50 e 200. Estas instâncias são *benchmarks* padrão para o PARG. Contudo, soluções ótimas para todas elas foram recentemente obtidas em [1]. As instâncias MGRAPH são, na realidade, instâncias do k-PARG para $k = 4$ ou $k = 5$.

O segundo conjunto, denominado R, contém instâncias do k-PARG com custos randômicos, geradas exatamente como sugerido em Caccetta e Hill [15]. As instâncias no conjunto R possuem $|V| \in \{100, 500, 800, 900, 1000\}$ e são portanto re-

presentativas das instâncias propostas em [15], onde a menor dimensão considerada é $|V| = 100$ e a maior é $|V| = 800$. Apresentamos apenas resultados para instâncias com $k = 3$ e $k = 4$ já que, por comparação, instâncias com $k = 5$ e $k = 6$ são bem mais fáceis de resolver. Para cada combinação possível de $|V|$, k e densidade de grafo p, onde $p \in \{0.05, 0.25, 0.5, 0.75, 1.0\}$, foram geradas 30 instâncias para $|V| \in \{100, 500, 800\}$ e 5 instâncias para $|V| \in \{900, 1000\}$ (em [15], 30 instâncias foram geradas para $|V| \leq 500$, enquanto que para $|V| \in \{600, 700, 800\}$, somente 5 instâncias foram geradas). Para todas as instâncias de R, os custos das arestas são escolhidos aleatoriamente no intervalo $[1, 1000]$, através de uma distribuição uniforme. Eventualmente, menos de 30 (para $|V| \in \{100, 500, 800\}$) ou menos de 5 (para $|V| \in \{900, 1000\}$) resultados são apresentados para algumas combinações de $|V|, k$ e p. Neste caso, resultados por ventura excluídos, são associados a grafos desconexos (ou seja, a instância correspondente é inviável para os dados de entrada utilizados) ou então associados a instâncias cujas soluções ótimas são trivialmente dadas por AGMs de G.

As instâncias no terceiro conjunto, denominadas D-E, foram geradas como sugerido em [1]. Para estas instâncias, pontos aleatoriamente gerados no plano Euclidiano são associados aos vértices de G e os custos das arestas são tomados como as distâncias Euclidianas entre as extremidades das mesmas, arredondando-se os valores obtidos para baixo. Para cada vértice (ponto no plano), coordenadas no plano Euclidiano são aleatoriamente escolhidas considerando-se uma distribuição uniforme no intervalo $[1, 1000]$. Adicionalmente, para cada instância no conjunto, os valores de b_i são escolhidos em $\{1, 2, 3\}$ (aleatoriamente, também considerando-se distribuição uniforme). Os grafos gerados são completos e o número de vértices correspondente a cada instância gerada é claramente identificado pelo nome da mesma. Três instâncias distintas foram geradas para cada $|V| \in \{100, 200, 300, 400, 500, 600\}$. Uma diferença entre as instâncias D-E aqui tratadas e aquelas sugeridas em [1] refere-se à freqüência de vértices i tais que $b_i = 1$. Em [1], estes vértices foram limitados a 10% de $|V|$. Aqui, a freqüência esperada destes vértices é de $\frac{1}{3}$. Outra diferença é que em [1], vértices com $b_i = 4$ são admitidos, enquanto que aqui, b_i é limitado a 3.

O quarto conjunto de instâncias testes foi gerado a partir das instâncias D-E. Para cada instância D-E, substituímos os custos Euclidianos por custos aleatórios escolhidos no intervalo $[1, 1000]$, utilizando uma distribuição uniforme. Este grupo de instâncias é denominado conjunto D-R.

4.4.1 Resultados computacionais

Os algoritmos propostos neste Capítulo foram programados em C e os experimentos computacionais (exceto aqueles apresentados na Tabela 4.4) foram efetuados em uma máquina Intel XEON com 3.06GHz. Já os resultados apresentados na Tabela 4.4 foram obtidos em uma máquina ATHLON AMD com 3GHz. O compilador GNU gcc foi utilizado (com chave de otimização -O3 ativada) no sistema operacional Linux, em ambos os casos.

Na Tabela 4.1, apresentamos os parâmetros de controle do algoritmo NDRC (veja Seção 2.3) empregados neste estudo computacional. A primeira coluna da Tabela indica o parâmetro de controle considerado. As demais colunas referem-se ao tipo de instância considerada. Cabe destacar que permitimos a realização de um número maior de ciclos do MS para as instâncias do conjunto R do que para as demais instâncias testadas. Esta decisão se baseou no fato de que, para as instâncias do conjunto R, os percentuais de fixação de variáveis são bastante elevados, logo nas primeiras iterações dos primeiros ciclos de MS. Assim sendo, se mostrou vantajoso interromper os primeiros ciclos iniciais do MS tão logo 50% das arestas do grafo de entrada do ciclo tivessem sido eliminadas. Desta forma, embora mais ciclos tenham sido empregados, o número de iterações por ciclo não foi exatamente o número máximo divulgado na última linha da Tabela 4.1. A seguir, apresentamos os resultados detalhados para cada conjunto de instâncias.

A Tabela 4.2 apresenta os resultados para as instâncias MGRAPH. As colunas naquela tabela indicam, respectivamente, para cada instância considerada, a identificação da mesma, o melhor limite inferior obtido por NDRC, denotado por z_d, o melhor limite superior, denotado por $\overline{z}$, o tempo de CPU (em segundos) necessário para se determinar estes valores (t), e os limites inferiores obtidos quando otimizamos sobre o politopo $P_1(n,b)$, ou seja $z_{SEC} = \min\left\{\sum_{e\in E} c_e x_e : x \in P_1(n,b)\right\}$ (obtido através de um algoritmo de Planos de Corte baseado na separação exata de desigualdades SECs [109], como descrito no Apêndice A). Deve ser observado que todas as instâncias no grupo MGRAPH são resolvidas na otimalidade com *pequeno esforço computacional*. Além disto, deve-se observar que os limites z_{SEC} são fortes para todos as instâncias no grupo.

Apesar de não serem exatamente as mesmas instâncias empregadas em [15], as instâncias no grupo R foram geradas exatamente como indicado naquela referência. Assim sendo, parece-nos razoável estabelecer comparações entre gaps de dualidade

Tabela 4.1: Parâmetros de controle empregados no algoritmo NDRC

	Grupo de Instâncias		
	MGRAPH	R	D-E e D-R
Sobreposição φ:	3%	1o. ciclo: 1% demais: 3%	3%
Freqüência de redução ξ:	50	60	$\|V\| \leq 300 : 50$ $\|V\| \geq 400 : 60$
No. máximo de ciclos de MS :	3	12	3
No. máximo de iterações de MS por ciclo :	1500	2000	$\|V\| \leq 300 : 1500$ $\|V\| \geq 400 : 2000$

médios (entre o custo das soluções ótimas e os correspondentes limites inferiores) obtidos em [15] e os aqui obtidos. Os resultados dessa comparação, para instâncias com $|V| \in \{100, 500, 800\}$ são apresentados na Tabela 4.3, onde as primeiras colunas indicam, respectivamente, $|V|$ (para cada grupo de instâncias), a densidade p esperada do grafo e o valor de k, na instância k-PARG correspondente. Nas colunas seguintes, são apresentados os limites Lagrangeanos divulgados em [15], expressos como gaps de dualidade médios a partir de valores ótimos correspondentes (o número entre parênteses se refere ao gap de dualidade para o algoritmo de Planos de Corte empregado em [15]). Uma referência (-) significa um valor 0.0, (*) significa que não houve necessidade de recorrer ao algoritmo de Planos de Corte em [15] (já que as instâncias foram resolvidas no pré-processamento), e (NR) significa que a correspondente informação não é disponível, uma vez que o tempo máximo de computação imposto em [15] (2 horas de computação em uma estação SUN SPARC II operando a 28.5 MIPS) foi excedido. Nas próximas cinco colunas, apresentamos os resultados relativos ao algoritmo NDRC. A primeira destas colunas fornece os gaps de dualidade máximo e médio entre os limites inferiores e superiores obtidos. Assim sendo, nossos gaps representam um limite superior para o gap que eventualmente obteríamos se valores ótimos fossem empregados, como em [15]. Nas últimas três colunas, indicamos, para cada conjunto de instâncias, os tempos médios de CPU

Tabela 4.2: Limites NDRC para as instâncias MGRAPH.

Instância	z_d	$\overline{z}$	t(s)	z_{SEC}
R050n1_4	4328.000	4328	0.06	4328.000
R050n2_4	4260.000	4260	0.05	4260.000
R050n3_4	4236.000	4236	0.07	4236.000
R100n1_4	8057.000	8057	0.22	8057.000
R100n2_4	8057.000	8057	0.36	8057.000
R100n3_4	8017.000	8017	0.27	8017.000
R200n1_4	16093.000	16093	3.17	16093.000
R200n2_4	15688.000	15688	4.24	15688.000
R200n3_4	15625.000	15625	3.88	15625.000
M050n1_5	6601.000	6601	0.08	6601.000
M050n2_5	5777.000	5777	0.10	5777.000
M050n3_5	5501.000	5501	0.10	5501.000
M100n1_5	11082.000	11082	0.74	11082.000
M100n2_5	11332.000	11332	1.54	11332.000
M100n3_5	10191.000	10191	0.57	10191.000
M200n1_5	18334.000	18334	3.04	18334.000
M200n2_5	19159.000	19159	4.18	19159.000
M200n3_5	16127.000	16127	3.80	16127.000

(em segundos), o número de instâncias resolvidas com garantia de otimalidade e o total de instâncias não triviais testadas. Finalmente, uma indicação (-), em qualquer coluna da Tabela 4.3 referente a um limite NDRC, corresponde ao valor zero.

Como pode ser observado na Tabela 4.3, os limites inferiores NDRC são mais fortes que os corresponentes limites de Relaxação Lagrangeana de [15] e são competitivos com os limites obtidos pelo algoritmo de Planos de Corte daquela referência. Em particular, os gaps relativos aos nossos limites inferiores são menores ou iguais aos seus equivalentes em [15], para toda combinação de $|V|$, p e k testada em nossos experimentos (isto ocorre apesar do fato de que nossos gaps são apenas um limite superior para os valores que seriam eventualmente obtidos, se fossem computados a partir de soluções ótimas, como feito em [15]). Verificamos também que nossos limi-

Tabela 4.3: Comparação de pré-processamento: instâncias R, $|V| \in \{100, 500, 800\}$

			Pré-processamento Lagrangeano de [15]	Resultados NDRC				
$\|V\|$	p	k	Qualidade dos limites % de diferença ao ótimo	Qualidade de z_d % de diferença ao UB		t(s)	No. de Problemas	
			Média	Max	Média		Resolvidos	Testados
100	0.05	3	0.21 (-)	-	-	0.26	30	30
		4	0.05 (-)	-	-	0.07	30	30
	0.25	3	0.26 (0.01)	0.06	0.002	0.40	29	30
		4	0.05 (-)	-	-	0.07	28	28
	0.50	3	0.23 (0.01)	0.12	0.007	0.46	28	30
		4	0.07 (-)	-	-	0.08	28	28
	0.75	3	0.23 (-)	0.13	0.004	0.37	29	30
		4	0.09 (-)	-	-	0.06	30	30
	1.00	3	0.25 (-)	0.19	0.006	0.33	29	30
		4	0.03 (-)	-	-	0.07	27	27
500	0.05	3	0.18 (-)	0.02	0.0006	7.11	29	30
		4	0.01 (-)	-	-	2.36	30	30
	0.25	3	0.22 (-)	-	-	14.16	30	30
		4	0.03 (0.01)	-	-	3.01	30	30
	0.50	3	0.24 (0.01)	0.04	0.001	18.05	29	30
		4	0.03 (0.01)	-	-	2.32	30	30
	0.75	3	0.28 (-)	-	-	18.81	30	30
		4	0.05 (0.01)	-	-	2.31	30	30
	1.00	3	0.31 (0.01)	-	-	15.71	30	30
		4	0.05 (-)	-	-	2.37	30	30
800	0.05	3	0.30 (-)	0.01	0.0006	23.19	28	30
		4	- (*)	-	-	9.96	30	30
	0.25	3	0.25 (-)	-	-	58.48	30	30
		4	0.01 (0.02)	-	-	11.56	30	30
	0.50	3	0.22 (-)	0.03	0.002	94.66	28	30
		4	0.03 (*)	-	-	7.82	30	30
	0.75	3	NR (NR)	-	-	102.42	30	30
		4	0.04 (*)	-	-	8.60	30	30
	1.00	3	NR (NR)	0.06	0.002	113.79	29	30
		4	0.04 (*)	-	-	7.72	30	30

tes inferiores são às vezes melhores, às vezes piores e, na maioria das vezes tão fortes quanto aqueles obtidos pelo algoritmo de Planos de Corte de [15]. Uma prova adicional da qualidade dos limites do nosso algoritmo NDRC é que, de um total de 893 instâncias R consideradas, 881 foram resolvidas com garantia de otimalidade pelo algoritmo proposto. Adicionalmente, para as 12 instâncias onde a otimalidade não foi provada, gaps de dualidade entre limites inferiores e superiores nunca excederam 0.19%.

Na Tabela 4.4, apresentamos os resultados para as instâncias R com $|V| \in \{900, 1000\}$. Conforme mencionado anteriormente, as instâncias testadas em Cacceta e Hill [15] foram restritas a $|V| \leq 800$.

Tabela 4.4: Resultados para instâncias R, $|V| \in \{900, 1000\}$

$\|V\|$	p	k	No. de problemas		t(s)
			Resolvidos	Testados	
900	0.05	3	5	5	140.71
		4	5	5	38.96
	0.25	3	5	5	237.17
		4	5	5	39.52
	0.50	3	5	5	395.56
		4	5	5	50.97
	0.75	3	5	5	317.06
		4	5	5	49.89
	1.00	3	5	5	307.01
		4	5	5	56.24
1000	0.05	3	5	5	190.53
		4	5	5	53.15
	0.25	3	5	5	325.77
		4	5	5	55.46
	0.50	3	5	5	333.85
		4	5	5	59.12
	0.75	3	5	5	330.82
		4	5	5	71.49
	1.00	3	5	5	446.03
		4	5	5	69.99

Assim sendo, na Tabela 4.4, apresentamos apenas os resultados obtidos por nosso algoritmo NDRC. As três primeiras colunas dessa Tabela indicam $|V|$, a densidade

p do grafo e o valor correspondente k. Nas colunas seguintes, indicamos o número de problemas resolvidos e o número de problemas testados para cada combinação de $|V|, p$ e k. Na última coluna, apresentamos o tempo médio de CPU (em segundos) para resolver o conjunto de instâncias testado. Como pode ser verificado, todas as 100 instâncias testadas foram resolvidas com garantida de otimalidade.

Note que nenhuma instância do grupo R com $k = 4$ deixou de ser resolvida por nosso algoritmo NDRC. Um aspecto que pode despertar atenção, a primeira vista, é o maior percentual de instâncias resolvidas com garantia de otimalidade para $|V| \in \{900, 1000\}$ (se comparado aos percentuais de intâncias resolvidas na otimalidade para, por exemplo, $|V| = 800$, apresentados na Tabela 4.3). Acreditamos que este *aparente* melhor desempenho do algoritmo para as maiores instâncias do grupo R seja justificado pelo fato de termos gerado apenas 5 instâncias testes para as mesmas. Provavelmente, se um número maior de instâncias tivesse sido gerado (por exemplo, 30 instâncias como feito para $|V| \in \{100, 500, 800\}$), o percentual de instâncias resolvidas seria certamente mais representativo e, possivelmente, menor que aquele obtido para $|V| = 800$.

A Tabela 4.5 apresenta resultados para o grupo D-E. As colunas naquela tabela indicam, respectivamente, a identificação da instância, o limite z_{SEC}, o limite $z_{BLO} = \min\left\{\sum_{e \in E} c_e x_e : x \in P_2(n, b)\right\}$ (também obtido através de um algoritmo de Planos de Corte, separando SECs através do algoritmo do Apêndice A e desigualdades Blossom, separadas através do algoritmo exato proposto em [82], descrito no Apêndice B), o melhor limite inferior NDRC, z_d, o melhor limite superior, $\overline{z}$, o gap de dualidade $(\overline{z} - z_d)/z_d$ (expresso em valores percentuais), o percentual de arestas sub-ótimas eliminadas, o número de ciclos de MS realizados, o correspondente número total de iterações realizados em cada ciclo (se no ciclo de MS a otimalidade não foi provada, ou caso cotrário, no último ciclo de MS), e o total de tempo de CPU gasto. Certificados de otimalidade foram obtidos para 2 das 18 instâncias testadas e um gap de dualidade médio (entre os melhores limites inferiores e superiores) de 0.82% foi observado. Os limites z_d obtidos por nosso algoritmo NDRC, que envolvem desigualdades Blossom, são sistematicamente mais fortes que z_{SEC} (dados pela Relaxação Linear de (3.1), quando somente SECs são usadas). De fato, isto ocorreu para todas as instâncias no conjunto, sendo a instância de_500_2 a única exceção. Estes resultados indicam claramente que a nossa heurística simples de separação conseguiu identificar desigualdades Blossom relevantes para fortalecer os limites in-

feriores. Além disto, para aquelas instâncias do conjunto D-E onde certificados de otimalidade não foram obtidos, na média, 83.5% das arestas do grafo foram eliminadas. Claramente, nossos limites z_d dominam aqueles obtidos em [1] (que no máximo seriam iguais a z_{SEC}), para instâncias geradas de forma similar.

Tabela 4.5: Limites NDRC: instâncias D-E.

Instâncias	z_{SEC}	z_{BLO}	z_d	$\overline{z}$	Gap %	% de AF	Ciclos - Itrs	t(s)
de_100_1	8775.000	8779.000	8778.410	8779	Opt	-	2-354	4.08
de_100_2	9623.000	9629.000	9629.000	9629	Opt	-	1-478	3.52
de_100_3	10783.000	10791.000	10783.499	10824	0.38	92.1	3-1500	13.19
de_200_1	12022.000	12029.000	12029.000	12048	0.16	97.8	3-1500	35.97
de_200_2	12611.500	12624.333	12617.388	12728	0.88	89.6	3-1500	71.69
de_200_3	12208.500	12224.000	12209.750	12325	0.94	89.0	3-1500	64.61
de_300_1	14758.500	14778.250	14769.219	14876	0.72	91.3	3-1500	159.65
de_300_2	13906.000	13927.000	13920.747	13931	0.07	98.8	3-1500	64.56
de_300_3	14920.500	14981.792	14940.565	15114	1.16	84.3	3-1500	168.96
de_400_1	19227.000	19239.000	19231.396	19388	0.81	87.0	3-2000	710.26
de_400_2	17373.500	17405.500	17384.581	17570	1.07	85.5	3-2000	615.32
de_400_3	16063.000	16074.500	16071.048	16099	0.17	97.9	3-2000	266.25
de_500_1	19332.500	19356.000	19345.037	19416	0.37	95.7	3-2000	659.41
de_500_2	22394.000	22397.000	22389.598	22749	1.61	58.2	3-2000	2513.00
de_500_3	19365.500	19394.750	19376.267	19743	1.89	64.4	3-2000	1400.76
de_600_1	21900.375	21917.677	21906.414	22248	1.56	67.4	3-2000	3050.76
de_600_2	21390.875	21424.500	21408.757	21679	1.26	76.3	3-2000	2258.23
de_600_3	22217.000	22239.000	22221.693	22610	1.75	61.1	3-2000	2760.85

A Tabela 4.6 mostra, para o conjunto de instâncias D-E, os resultados correspondentes ao *warm start* para um algoritmo de Planos de Corte, como sugerido na Seção 4.3. As colunas naquela tabela indicam, respectivamente, a identificação da instância, o limite z_{SEC}, o limite z_{BLO}, o melhor limite inferior obtido pelo algoritmo NDRC, z_d, e dois limites inferiores avançados, z_{W_1} e z_{W_2}. Para o limite z_{W_1}, considera-se apenas as desigualdades SECs identificadas pelo procedimento introduzido na Seção 4.3 (BIs não são empregadas). Para o limite z_{W_2}, as desigualdades

Blossom e SECs identificadas na Seção 4.3 são utilizadas. Como pode ser observado, das 16 instâncias consideradas na Tabela 4.6, em 12 casos os limites z_{W_2} são mais fortes do que os melhores limites inferiores correspondentes obtidos pelo algoritmo NDRC. Para as outras 4 instâncias, os limites z_{W_2} equivalem a, no mínimo, 99.9% do correspondente limite inferior z_d. Além disto, os limites z_{W_1} são muito próximos aos valores correspondentes a z_{SEC}. Estes resultados claramente validam o algoritmo NDRC como uma abordagem efetiva para gerar limites inferiores bastante fortes para o PARG. Os resultados também validam o procedimento de identificação de SECs sugerido na Seção 4.3.

Tabela 4.6: Limites avançados: instâncias D-E.

Instância	z_{SEC}	z_{BLO}	z_d	Limites avançados	
				z_{W_1}	z_{W_2}
de_100_3	10783.000	10791.000	10783.499	10783.000	10783.500
de_200_1	12022.000	12029.000	12029.000	12022.000	12029.000
de_200_2	12611.500	12624.333	12617.388	12611.500	12617.917
de_200_3	12208.500	12224.000	12209.750	12208.500	12210.500
de_300_1	14758.500	14778.250	14769.219	14758.500	14770.000
de_300_2	13906.000	13927.000	13920.747	13906.000	13912.500
de_300_3	14920.500	14981.792	14940.565	14920.500	14943.417
de_400_1	19227.000	19239.000	19231.396	19198.000	19224.000
de_400_2	17373.500	17405.500	17384.581	17373.500	17386.625
de_400_3	16063.000	16074.500	16071.048	16061.000	16072.000
de_500_1	19332.500	19356.000	19345.037	19332.167	19340.75
de_500_2	22394.000	22397.000	22389.598	22374.330	22382.500
de_500_3	19365.500	19394.750	19376.267	19365.500	19381.250
de_600_1	21900.375	21917.677	21906.414	21900.375	21907.100
de_600_2	21390.875	21424.500	21408.757	21390.375	21409.375
de_600_3	22217.000	22239.000	22221.693	22215.500	22222.976

A Tabela 4.7 apresenta resultados para o conjunto de instâncias D-R. A descrição das colunas nesta tabela é similar àquela apresentada para a Tabela 4.5. Das 18 instâncias testadas, obteve-se certificados de otimalidade em 4 casos e um gap de

Tabela 4.7: Limites NDRC: instâncias D-R.

Instância	z_{SEC}	z_{BLO}	z_d	$\overline{z}$	Gap %	% de AF	Ciclos - Itrs	t(s)
dr_100_1	2174.750	2177.231	2176.731	2197	0.93	95.9	3-1500	6.52
dr_100_2	2674.000	2674.000	2673.971	2674	Opt	-	1-713	4.39
dr_100_3	2689.000	2689.000	2688.988	2689	Opt	-	3-1015	7.07
dr_200_1	2139.000	2139.000	2138.999	2148	0.42	97.9	3-1500	36.58
dr_200_2	2469.875	2470.250	2469.850	2482	0.49	97.7	3-1500	44.27
dr_200_3	2402.750	2402.750	2402.663	2409	0.26	98.3	3-1500	36.40
dr_300_1	2460.000	2460.000	2459.933	2476	0.65	97.7	3-1500	140.44
dr_300_2	2312.000	2312.000	2311.990	2312	Opt	-	3-1500	85.52
dr_300_3	2583.722	2584.125	2583.862	2621	1.44	95.7	3-1500	198.32
dr_400_1	2814.667	2815.017	2814.023	2944	4.62	87.6	3-2000	638.92
dr_400_2	2441.034	2441.034	2440.320	2483	1.75	95.5	3-2000	503.46
dr_400_3	2387.000	2387.000	2386.948	2387	Opt	-	3-400	155.13
dr_500_1	2595.625	2595.625	2595.597	2604	0.32	99.4	3-2000	549.88
dr_500_2	2650.000	2650.000	2649.999	2814	6.19	84.8	3-2000	1340.40
dr_500_3	2591.851	2591.884	2591.712	2607	0.59	98.1	3-2000	713.20
dr_600_1	2819.519	2819.571	2819.460	2857	1.33	98.1	3-2000	1271.31
dr_600_2	2659.668	2659.668	2659.647	2699	1.48	96.0	3-2000	1705.37
dr_600_3	2798.452	2798.468	2798.298	2828	1.06	96.9	3-2000	1075.67

dualidade médio (entre limites inferiores e superiores) de 1.20% (valor altamente influenciado pelo gap obtido para as instâncias dr_400_1 e dr_500_2, de cerca de 4% e 6%, respectivamente). Como pode ser observado, para as instâncias neste grupo, os limites z_{BLO} (que consideram desigualdades Blossom) superam os limites z_{SEC} (quando apenas SECs são separadas) em 5 casos. Comparando com as instâncias D-E, observa-se que o ganho de z_{BLO} em relação a z_{SEC} foi bem menos expressivo. Para o caso da instância dr_100_3, apesar dos valores z_{BLO} e z_{SEC} serem iguais, a solução da Relaxação Linear da formulação baseada em desigualdades Blossom é inteira. Tendo em mente a pequena diferença entre z_{BLO} e z_{SEC} para estas instâncias, podemos dizer também que os limites z_d foram bastante fortes, apesar de terem superado z_{SEC} em apenas duas instâncias (dr_100_1 e dr_300_3).

Uma possível explicação para o menor impacto das desigualdades Blossom no fortalecimento dos limites para as instâncias do grupo D-R é a seguinte. Para que uma desigualdade Blossom seja violada por um ponto extremo $\overline{x} \in P_1(n,b)$ é necessário que haja uma certa competição em custo entre as arestas que induzem esta desigualdade. Na verdade, se algumas arestas nesta desigualdade tiverem custos muito maiores que as demais, a desigualdade tende a ser pouco restritiva em relação a $P_1(n,b)$. Diante de custos uniformemente distribuidos no intervalo $[1,1000]$ e de $|V| \leq 600$, para as instâncias D-R, muitas das arestas incidentes a um dado vértice são naturalmente pouco competitivas. Para que houvesse maior competição entre elas, a ordem de grandeza do número de arestas incidentes a um vértice (limitado por $|V|$) deveria ser bem maior que o limite superior de custos empregado (1000 no caso testado). Esta observação é experimentalmente confirmada pelos elevados índices de fixação de variáveis obtidas para as as instâncias do grupo D-R, apesar de possuirem gaps de dualidade mais elevados (em relação àqueles correspondentes às instâncias D-E).

Finalmente, a Tabela 4.8 apresenta, para as instâncias D-R, resultados obtidos pelo procedimento de warm-start sugerido na Seção 4.3. Os dados nesta Tabela fornecem informações similares àqueles apresentados na Tabela 4.6. De um modo geral, os resultados da Tabela 4.8 indicam que o procedimento de warm start proposto parece funcionar muito bem. Isto pode ser comprovado pelo fato de que os limites induzidos por ele são melhores ou iguais aos correspondentes limites inferiores NDRC. Em síntese, o procedimento de warm-start sugerido se comportou de forma bastante desejável.

Tabela 4.8: Limites avançados: instâncias D-R.

Instância	z_{SEC}	z_{BLO}	z_d	Limites avançados z_{W_1}	Limites avançados z_{W_2}
dr_100_1	2174.750	2177.231	2176.731	2174.750	2174.750
dr_200_1	2139.000	2139.000	2138.999	2139.000	2139.000
dr_200_2	2469.875	2470.250	2469.850	2469.857	2469.857
dr_200_3	2402.750	2402.750	2402.663	2402.750	2402.750
dr_300_1	2460.000	2460.000	2459.933	2459.982	2459.982
dr_300_3	2583.722	2583.862	2583.862	2583.722	2583.722
dr_400_1	2814.667	2815.017	2814.023	2813.598	2814.071
dr_400_2	2441.034	2441.034	2440.320	2440.350	2440.942
dr_400_3	2387.000	2387.000	2386.948	2387.000	2387.000
dr_500_1	2595.625	2595.625	2595.597	2595.625	2595.625
dr_500_2	2650.000	2650.000	2649.999	2650.000	2650.000
dr_500_3	2591.851	2591.884	2591.712	2591.743	2591.743
dr_600_1	2819.519	2819.571	2819.460	2819.500	2819.500
dr_600_2	2659.668	2659.668	2659.647	2659.667	2659.667
dr_600_3	2798.452	2798.468	2798.298	2798.382	2798.382

Avaliando os resultados do algoritmo NDRC para os diversos conjuntos de instâncias considerados, afirmamos que as instâncias k-PARG sugeridas em Caccetta e Hill [15] são muito mais fáceis de se resolver que as instâncias Euclidianas D-E. São também mais fáceis de se resolver que as instâncias com custos aleatórios D-R. Isto pode ser observado se comparamos o número de instâncias resolvidas na otimalidade (98.8% para instâncias R contra 11.1% para instâncias D-E) ou o tempo de CPU associados a cada caso. Observe que ao comparar as instâncias D-E com $|V| = 500$ com as instâncias R com $|V| = 500$, $k = 3$ e $p = 1.0$, verificamos que o algoritmo NDRC é 100 vezes mais rápido para o último conjunto. Devemos também salientar que o tempo exigido pelo algoritmo para resolver na otimalidade, com uma única exceção, todas as 30 instâncias do grupo R com $|V| = 800$, $k = 3$ e $p = 1.00$ é comparável ao tempo exigido pelo mesmo algoritmo para obter um gap de dualidade de 1.61% para a instância de_600_1, sozinha !

4.5 Comentários finais

Neste capítulo, pela primeira vez na história do PARG, fortalecemos os limites inferiores dados pela formulação clássica (3.1) através do uso de desigualdades Blossom. Isso foi feito através de um algoritmo do tipo NDRC. A idéia central do algoritmo consiste em dualizar, de forma Lagrangeana, desigualdades Blossom na medida em que são violadas pela solução do PL(λ) corrente. Para resolver o problema de separação associado, propusemos uma heurística gulosa que, apesar de sua simplicidade, conseguiu identificar desigualdades relevantes para, na prática, obter limites mais fortes que aqueles alcançados pela formulação clássica (3.1).

Os experimentos computacionais conduzidos aqui indicaram que os limites inferiores obtidos ao se incorporar desigualdades Blossom dominam ou são competitivos com os limites que consideram apenas as SECs. Apesar de uma comparação direta não ter sido realizada, os limites superiores apresentados aqui parecem ser competitivos (em termos de tempo de computação e qualidade das soluções) aos melhores disponíveis na literatura (veja [1, 114]). Em particular, algumas instâncias *benchmark* para o PARG foram resolvidas na otimalidade, com certa facilidade. Além disto, conduzimos experimentos computationais com um conjunto de instâncias gerados como proposto em [1]. Para este grupo de instâncias, a qualidade dos nossos limites superiores é certamente tão boa quanto aquela alcançada em [1] (para instâncias de dimensões semelhantes) uma vez que nossa heurística Lagrangeana se beneficia de informação dual de melhor qualidade, oriunda de uma formulação mais forte.

Uma contribuição adicional deste Capítulo consiste em um procedimento para transportar limites inferiores Lagrangeanos para um algoritmo de Planos de Corte, sem a necessidade de recorrer a algoritmos de separação. Embora a idéia de um procedimento de warm-start, fazendo uso de desigualdades identificadas em um algoritmo Relax-and-Cut não seja nova (veja [16]), o procedimento aqui proposto é mais elaborado que aquele sugerido em [16]. Isto é verdade uma vez que, ao invés de simplesmente utilizarmos as desigualdades Blossom identificadas pelo algoritmo NDRC em uma Relaxação Linear Inicial para o PARG, também tentamos caracterizar o ponto extremo do politopo das árvores geradoras associado ao melhor limite NDRC obtido. Assim sendo, o procedimento proposto aqui pode ser adaptado a qualquer problema de Otimização Combinatória no qual os Problemas Lagrangea-

nos definam matróides.

No próximo Capítulo, apresentaremos um algoritmo híbrido exato, do tipo Branch-and-Cut para o PARG. Nesse algoritmo, como um módulo de pré-processamento, utilizamos o algoritmo NDRC aqui proposto. Os resultados descritos neste Capítulo foram publicados em [28].

Capítulo 5

Um algoritmo híbrido Relax-and-Cut / Branch-and-Cut para o PARG

Neste Capítulo, apresentamos um algoritmo híbrido para a resolução exata do PARG. O algoritmo é composto por duas etapas: uma fase de pré-processamento (baseada no algoritmo NDRC descrito no Capítulo anterior) e um algoritmo Branch-and-Cut no qual cutsets, SECs e desigualdades Blossom são separadas.

5.1 Introdução

Na literatura, o único algoritmo do tipo Branch-and-Cut conhecido para resolver o PARG foi proposto por Caccetta e Hill [15]. Naquela referência, em uma fase de pré-processamento, os autores utilizam uma heurística Lagrangeana (baseada em um procedimento de ajuste de multiplicadores, proposto em Volgenant [122]), para obter limites primais iniciais e, com a ajuda de limites duais, eliminar arestas subótimas do problema. A seguir, um algoritmo Branch-and-Cut baseado na separação heurística de SECs e cutsets é aplicado.

Neste Capítulo, propomos um novo algoritmo exato para o PARG. O algoritmo envolve duas etapas: na primeira, o algoritmo NDRC descrito no Capítulo 4 é utilizado em um pré-processamento das instâncias; na segunda, um novo algoritmo Branch-and-Cut, distinto daquele em [15], é aplicado às instâncias pré-processadas. Em uma avaliação preliminar, poder-se-ia dizer que o algoritmo proposto aqui assemelha-se ao de Caccetta e Hill [15]. Entretanto, como veremos adiante, em suas duas fases, nosso algoritmo contém ingredientes que o diferenciam bastante do algoritmo em [15]. Em linhas gerais, o nosso algoritmo se diferencia por empregarmos desigualdades Blossom tanto na fase NDRC quanto na fase de Planos de

Corte. Se diferencia também por utilizarmos uma heurística primal que contém uma busca local, ao longo de toda a árvore de enumeração. No entanto, provavelmente, a diferença mais marcante entre os dois algoritmos reside no fato de utilizarmos limites inferiores gerados pelo algoritmo NDRC para uma inicialização avançada, sem necessidade de resolver problemas de separação, de um algoritmo de Planos de Corte.

Este Capítulo é organizado da seguinte forma. Na Seção 5.2, apresentamos as características principais do algoritmo Branch-and-Cut que propusemos, dando ênfase às estratégias para a geração e gerenciamento de cortes. Na Seção 5.3, apresentamos os resultados computacionais obtidos pelo algoritmo híbrido Relax-and-Cut / Branch-and-Cut. Encerramos o Capítulo, na Seção 5.4, com alguns comentários e sugestões para investigações futuras.

5.2 O algoritmo Branch-and-Cut

À seguir, detalhamos a implementação das estratégias de geração de cortes, obtenção de limites e sistemas de controle do algoritmo Branch-and-Cut. Uma vez que a etapa de pré-processamento desse algoritmo é feita através do algoritmo NDRC, este aspecto não será novamente abordado neste Capítulo.

5.2.1 Desigualdades violadas

Uma Relaxação Linear inicial para o PARG, considerada em nosso algoritmo Branch-and-Cut, é dada por

$$z_{PL} = \min \left\{ \sum_{e \in E} c_e x_e : x \in \mathbb{R}^{|E|} \cap P \right\}, \tag{5.1}$$

onde a região poliedral P é definida pela interseção das desigualdades (3.2), (3.4) e (3.5). Seja $\overline{x}$ a solução ótima de (5.1) e $\overline{G} = (V, \overline{E})$ o subgrafo por ela induzido em G, onde $\overline{E} = \{e \in E : 0 < \overline{x}_e \leq 1\}$. Se $\overline{x}$ é inteira e $\overline{G}$ é conexo, $\overline{x}$ resolve (3.1). Caso contrário, tentamos indentificar desigualdades válidas (ou cortes) violadas por $\overline{x}$ que, adicionados às restrições que definem P, geram uma nova região poliedral P, mais restrita, e fortalecem os limites inferiores dados por z_{PL}. O nosso algoritmo Branch-and-Cut baseia-se na separação de três classes de desigualdades válidas para o PARG: cutsets (3.6), SECs (3.3) e desigualdadades Blossom (3.10). A seguir,

detalhamos as estratégias empregadas para a separação destas desigualdades, de forma exata ou heurística.

5.2.1.1 A separação de desigualdades cutsets

É sabido que uma formulação não direcionada para encontrar árvores geradoras, baseada em cutsets, é mais fraca que uma formulação (também não direcionada) baseada em SECs (veja [94], por exemplo). Apesar disto, o emprego de desigualdades cutsets em um algoritmo Branch-and-Cut pode ser atraente, se conduzido adequadamente. Uma razão para isto é o fato de que a separação exata de cutsets pode ser realizada em tempo $O(|V|^3)$, enquanto a separação exata de SECs consome, no pior caso, $O(|V|^4)$.

A idéia central da separação das desigualdades cutsets é tentar obter, caso exista, um conjunto de vértices $S \subset V$ tal que a capacidade do corte $C(S, V \backslash S)$ em $\overline{G}$, isto é, $\sum_{(i,j) \in \overline{E}: i \in S, j \notin S} \overline{x}_{(i,j)}$, seja inferior à unidade. Em caso de sucesso, uma desigualdade cutset violada foi obtida. Em outras palavras, a separação de cutsets equivale a resolver um problema de corte mínimo global em uma rede capacitada definida a partir de $\overline{G}$ e de $\overline{x}$. Cada aresta $e \in \overline{E}$ define uma aresta na rede, com capacidade $\overline{x}_e$.

Uma alternativa para resolver, mais rapidamente, a separação de desigualdades cutsets é o emprego de procedimentos de contração [106]. Tais procedimentos permitem obter, a partir de $\overline{G}$, um grafo reduzido $\overline{G}_r = (\overline{V}_r, \overline{E}_r)$. A contração baseia-se na fusão das extremidades de arestas (i, j) tais que $\overline{x}_{(i,j)} = 1$. Veja que, neste caso, um corte que separe i de j não pode gerar uma desigualdade cutset violada, já que sua capacidade é, no mínimo, igual a 1. Portanto, i e j devem pertencer ao mesmo conjunto de um corte que induz uma desigualdade violada. Suponhamos agora que i e j possuem um vértice adjacente comum, digamos, o vértice l, no grafo $\overline{G}$. Neste caso, a fusão de i e j em um novo vértice $\{i, j\}$ implica também na fusão das arestas (i, l) e (j, l) em uma nova aresta $(\{i, j\}, l)$. Naturalmente, nesse caso, é necessário tomar $\overline{x}_{(\{i,j\},l)} = \overline{x}_{(i,l)} + \overline{x}_{(j,l)}$. Note que, ao corrigirmos os valores de $\overline{x}_e$ para as arestas que sofreram fusão, há a preservação da capacidade dos cortes entre os vértices da rede definida a partir de $\overline{G}$, antes e após a fusão.

O procedimento de contração é aplicado recursivamente, até o ponto em que não mais existem arestas tais que $\overline{x}_e \geq 1$. Assim sendo, a rede definida a partir do grafo reduzido $\overline{G}_r$, obtido ao final do procedimento de contração, é equivalente, em termos

de fluxo, à rede definida a partir do grafo original $\overline{G}$. Portanto, para identificar desigualdades cutsets violadas, podemos aplicar um algoritmo que determine um corte mínimo global na rede definida a partir de $\overline{G}_r$ ao invés de aplicar o mesmo algoritmo na rede definida a partir de $\overline{G}$. Em nossa implementação da separação de desigualdades cutsets, empregamos o algoritmo de Hao e Orlin [65] para determinar o corte mínimo global. Dada uma rede capacitada com n vértices, o algoritmo de [65] obtém o corte mínimo global em tempo $O(n^3)$.

Normalmente, em muitas Relaxações Lineares dos subproblemas PI_i resolvidos ao longo da árvore de enumeração, muitas arestas $e \in \overline{E}$ satisfazem à condição $\overline{x}_e = 1$. Na medida em que duas arestas são fundidas em uma, mais arestas podem vir a satisfazer a condição $\overline{x}_e \geq 1$. Desta forma, as dimensões do grafo final $\overline{G}_r$ são muitíssimo reduzidas se comparadas às dimensões de $\overline{G}$. Consequentemente, as dimensões das redes defindas a partir de $\overline{G}_r$ são bem inferiores às dimensões das redes definidas a partir de $\overline{G}$. Como conseqüência disso, o algorimo de Hao e Orlin pode se beneficiar enormemente do procedimento de contração.

Note que o procedimento de contração oferece, como sub-produto, uma heurística para a separação de SECs: se $k = \{j_1, ..., j_p\} \subset V$ representa um vértice em $\overline{V}_r$ e se $x(E(\{j_1, ..., j_p\})) > p-1$, uma SEC violada foi encontrada. Esta heurística também é empregada em nossa implementação.

Eventualmente, o grafo $\overline{G}$ pode ser desconexo. Assim sendo, para cada componente conexo de $\overline{G}$, aplicamos o procedimento de contração descrito e o algoritmo de Hao e Orlin. Caso um corte mínimo global com capacidade inferior à unidade seja encontrado, o mesmo é inserido no cut-pool e na formulação do problema. Caso contrário, os vértices de cada componente conexa induzem uma cutset violada. Desta forma, mais de uma desigualdade cutset violada pode ser obtida ao se aplicar nosso algoritmo de separação. Nos experimentos computacionais efetuados, a inserção de mais de uma desigualdade violada por vez, para esta classe de desigualdades válidas, se mostrou vantajosa.

5.2.1.2 Separação exata de desigualdades SEC

O procedimento de contração, embora muito importante para a separação de cutsets, pode fazer com que, após as operações de fusão, desigualdades SECs violadas no grafo original $\overline{G}$ não sejam identificadas no grafo resultante $\overline{G}_r$. Por este motivo, em certas situações a serem posteriormente descritas, aplicamos também o algo-

ritmo de Padberg e Wolsey [109] (veja o Apêndice A) ao grafo original $\overline{G}$. Neste caso, dentre as desigualdades obtidas ao longo da execução do algoritmo em [109], inserimos na Relaxação Linear corrente do PARG aquela desigualdade cuja violação é máxima. Em caso de empate, damos preferência à desigualdade mais esparsa. Em nossos experimentos computacionais, observamos que a estratégia de introduzir uma única desigualdade SEC por vez (a mais violada) produziu resultados superiores (em termos de tempo de CPU) que outras estratégias usualmente empregadas, como por exemplo, a introdução de vários cortes violados por vez (ordenados por violação) [108] ou a introdução de cortes ortogonais [92]. Em parte, acreditamos que este comportamento pode ser explicado pelo fato de que algumas desigualdades SECs encontradas pelo algoritmo de [109] diferem por poucos vértices. Neste caso, a aplicação de várias rodadas de cortes ortogonais deveria, em princípio, ser vantajosa (veja [92]). Entretanto, para este problema em particular, o custo adicional de avaliar o corte ortogonal mais violado, recalcular o grafo $\overline{G}$ e prosseguir com a separação de novas desigualdades SECs, não compensou o ganho obtido nos limites duais.

Conforme exposto no Capítulo 3, uma SEC definida por um conjunto S que contém algum vértice i tal que $b_i = 1$ não pode definir uma faceta para o politopo $PARG(n, b)$. Além disto, diante da satisfação das desigualdades (3.4), se S gera uma SEC violada, então $S \setminus \{i\}$ também gera. Podemos então eliminar os vértices i tais que $b_i = 1$ dos conjuntos obtidos com os procedimentos de separação, sem comprometer a violação da desigualdade SEC associada. Na aplicação do algoritmo em [109], podemos ir um pouco além. Para a determinação da rede N, descrita no Apêndice A, desconsideramos todos os vértices i (e as arestas a eles incidentes) tais que $\overline{x}(\delta(i)) \leq 1$. Conforme salientado em [109], esta redução deve ser aplicada em um único passe, não podendo, portanto, ser aplicada recursivamente. Isto se justifica porque a aplicação sucessiva do procedimento pode permitir que desigualdades SECs violadas na rede original não sejam caracterizadas na rede resultante.

5.2.1.3 Separação heurística de desigualdades Blossom

Embora o algoritmo de Letchford et al. [82] (LRT), descrito em detalhes no Apêndice B, resolva de forma exata a separação de desigualdades Blossom com complexidade idêntica à separação exata das SECs ($O(|V|^4)$), este procedimento não foi empregado no nosso algoritmo Branch-and-Cut. O uso do algoritmo LRT foi restrito à

obtenção dos limites z_{BLO}, apresentados no Capítulo anterior. Isto porque aquele algoritmo se mostrou, na prática, lento e também porque a heurística de separação de desigualdades Blossom que propusemos foi capaz de fechar praticamente toda a distância entre z_{BLO} e z_{SEC}, em tempos computacionais bastante aceitáveis. Assim sendo, um equilíbrio satisfatório foi alcançado entre o fortalecimento dos limites inferiores para o PARG (através do uso de desigualdades Blossom) e o custo adicional dispendido para identificá-las. Dessa maneira, nos resultados computacionais que se seguem, a heurística de separação mencionada acima foi o único instrumento utilizado para a identificação de desigualdades Blossom violadas. Esta heurística é apresentada no Algoritmo 5.1 abaixo.

Algoritmo 5.1 Heurística de Separação de desigualdades Blossom, no algoritmo Branch-and-Cut para o PARG

Determine $\overline{G} = (V, \overline{E})$, o grafo induzido pela solução $\overline{x}$ que se deseja separar.
para todo $i \in V$ **faça**
    $s'_i = \overline{x}(\delta(i)) - b_i$
    $b'_i = b_i$
fim para
$E' \leftarrow \overline{E}$
para todo $(i,j) \in E' : \overline{x}_{(i,j)} = 1$ **faça**
    $E' \leftarrow E' \setminus \{(i,j)\}$
    $b'_i \leftarrow b'_i - 1$
    $b'_j \leftarrow b'_j - 1$
fim para
Determine os p componentes conexos $G'_i = (V'_i, E'_i), i = 1, ..., p$, do grafo $G' = (V, E')$.
para todo $i \in \{1, ..., p\}$ **faça**
    $s'(V'_i) := \sum_{j \in V'_i} s'_j$
    $b'(V'_i) := \sum_{j \in V'_i} b'_i$
    $T(V'_i) := \{e \in \delta(V'_i) \subset \overline{E} : \overline{x}_e = 1\}$
    Se $((s'(V'_i) < 1)$ e $(b'(V'_i)$ é ímpar$))$ **então**
        Introduza a desigualdade violada gerada pelo handle V'_i e tooth $T(V'_i)$
    fim Se
fim para

Inicialmente, no Algoritmo 5.1, eliminamos do grafo induzido pela solução $\overline{x}$ a ser separada, todas as arestas e para as quais $\overline{x}_e = 1$. A eliminação destas arestas leva o grafo resultante G' a ter, eventualmente, mais de uma componente conexa. Uma vez determinadas estas componentes, se a folga $s'(V'_i)$ associada à componente $G_i = (V'_i, E'_i)$ for inferior à unidade e se G_i for ímpar (isto é, se $b'(V'_i)$ for ímpar), a desigualdade gerada por $V'_i, T(V'_i)$ é garantidamente violada. Isto pode

ser confirmado ao se observar a desigualdade blossom na forma

$$x(\delta(H) \setminus T) + \sum_{e \in T \subseteq \delta(H)} (1 - x_e) + \sum_{i \in H} (b_i - x(\delta(i)) \geq 1. \tag{5.2}$$

Baseado na Observação 3.4 apresentada no Capítulo 3, por fim, empregamos um procedimento de *lifting* para as desigualdades Blossom obtidas através da heurística acima. Suponha que o par H, T gere uma desigualdade violada. Se para um dado vértice $i \in V, i \notin H$, a condição $|\{e : e \in (T \cap \delta(i)), \overline{x}_e = 1\}| = b_i$ for satisfeita, podemos gerar uma desigualdade mais forte através da substituição de H e T por $H_i = H \cup \{i\}$ e $T_i = T \setminus \{e : e \in (T \cap \delta(i)), \overline{x}_e = 1\}$, respectivamente. Note que o termo independente da desigualdade gerada por H_i, T_i é igual ao termo independente da desigualdade gerada por H, T. Note também que a nova desigualdade pode incluir arestas (ou seja, variáveis na clique induzida por H') ausentes na desigualdade original.

5.2.2 O gerenciamento do gerador de cortes e do cut-pool

Uma vez que trabalhamos com classes distintas de desigualdades e empregamos algoritmos diferentes para a identificação de desigualdades violadas, adotamos um esquema hieráquico para dar, ao longo do algoritmo, maior ou menor prioridade à chamada de alguns destes procedimentos. Nessa lógica, a hierarquia estabelecida teve por objetivo reduzir o tempo global de CPU envolvido. Tentamos alcançar este objetivo, estabelecendo um esquema no qual algoritmos de separação mais caros seriam chamados apenas quando:

- desigualdades violadas não foram encontradas no cutpool;
- algoritmos mais baratos falharam na identificação de desigualdades violadas;
- o crescimento do limite inferior no nó corrente ficou abaixo de um certo parâmetro de controle estabelecido (aqui denominado de MINIMP).

Em termos práticos, os procedimentos de geração de cortes e de controle de *tailing off* foram integrados da seguinte forma. Em cada nó da árvore de enumeração, a cada Relaxação Linear resolvida, tentamos identificar desigualdades violadas no cutpool. Estas desigualdades podem, eventualmente, ter sido identificadas pelo procedimento de warm-start proposto na Seção 4.3 ou, mais provavelmente, pelos algoritmos de separação apresentados. Se não encontrarmos desigualdades violadas

(de qualquer classe) no cutpool ou se o crescimento do limite dual naquela iteração for inferior a MINIMP chamamos o gerador de cortes. Através dele, tentamos inicialmente identificar cutsets e SECs violadas através da heurística descrita neste Capítulo. Caso a heurística não identifique desigualdades violadas ou, novamente, caso o crescimento do limite dual for inferior a MINIMP, recorremos ao procedimento de Padberg e Wolsey [109] (descrito no Apêndice A) e ao Algoritmo 5.1. Tudo isto só ocorre se, em pelo menos uma de MAXITER iterações consecutivas, o crescimento do limite dual for superior a MINIMP. Caso contrário, concluímos que não vale a pena continuar explorando o nó e, então, efetuamos sua ramificação. Eventualmente, a solução obtida após MAXITER iterações sem um crescimento mínimo desejado para o limite inferior, pode ser inteira e não conexa. Neste caso particular, o algoritmo de separação de cutsets é sempre chamado, e o subproblema é novamente otimizado.

Critérios para a manutenção de desigualdades inseridas nas matrizes associadas às Relaxações Lineares e no cut-pool são descritos a seguir. Em relação às desigualdades presentes numa matriz, adotamos a seguinte estratégia: sempre que houver um incremento igual ou superior a MINIMP no limite dual da última iteração de um algoritmo de Planos de Corte, eliminamos as desigualdades não ativas (folgadas) ou aquelas cujas variáveis de folga associadas sejam básicas. Note que a eliminação destas desigualdades não implica em perda de viabilidade da solução básica corrente. Dado que o procedimento de separação de cutsets é menos dispendioso que os demais procedimentos de separação sugeridos e que este é chamado com bastante freqüência ao longo do algoritmo, sempre que partirmos para a ramificação em um nó, eliminamos do cutpool as desigualdades cutsets não ativas, geradas naquele nó. As desigualdades Blossom e SECs são preservadas no cutpool por todo o algoritmo.

5.2.3 Detalhes de implementação

Na tentativa de obter melhores limites superiores para o PARG que aqueles previamente obtidos pelo algoritmo NDRC e antecipar assim a obtenção de soluções viáveis ao longo da árvore de enumeração, adaptamos os procedimentos primais descritos na Seção 4.2.2 para o algoritmo Branch-and-Cut. De forma análoga à empregada na heurística Lagrangeana, utilizamos custos complementares $\{c_e(1-\overline{x}_e) : e \in E\}$ como entrada para o algortimo construtivo descrito na Seção 4.2.2. Posteriormente, recorremos à busca local, lá descrita, sob os custos originais. Em função do baixo

custo computacional dos nossos procedimentos primais (em grande parte devido aos testes de dominância descritos na Seção 4.2.2), empregamos os procedimentos aqui descritos em cada iteração do algoritmo de Planos de Corte, em qualquer nó da árvore de enumeração.

Uma vez que pretendemos avaliar o desempenho do nosso algoritmo Branch-and-Cut híbrido em termos de suas contribuições principais, a saber, o uso de desigualdades Blossom e os ganhos em tempo de CPU obtidos pelo uso do procedimento de warm start descrito na Seção 4.3, adotamos regras usualmente empregadas na literatura para definir os critérios de ramificação e de seleção de nós.

Os demais aspectos importantes envolvidos na implementação do algoritmo Branch-and-Cut são:

- O resolvedor `XPRESS-MP`, versão `15.25`, foi utilizado com chaves de pré-processamento desligadas (exceto para fixação de variáveis através de precificação dual). Todas as opções que permitem a geração automática de cortes ou a busca de soluções viáveis através de heurísticas do próprio software foram desligadas.

- Dentre as estratégias de seleção de nós mais empregadas na literatura, testamos a *Busca em Profundidade* e a *Melhor Primeiro*. Não houve uma indicação clara de qual estratégia se comportou melhor nos experimentos computacionais que efetuamos. Para algumas instâncias, a estratégia *Melhor Primeiro* levou a menores tempos de CPU. Para outras, isso ocorreu com o uso da *Busca em Profundidade*. Os resultados computacionais apresentados na próxima Seção foram obtidos utilizando a estratégia *Busca em Profundidade*.

- O critério para a ramificação de nós aqui empregado tem por base as variáveis do problema. A variável de definição da ramificação é determinada automaticamente pelo resolvedor. Esta estratégia funciona da seguinte forma. Para cada variável candidata a definir a ramificação, é atribuída uma medida de variação do limite dual no nó em questão. Esta medida depende da média de uma estimativa do limite dual ao se fixar a variável em seus limites inferior e superior. Dentre as candidatas existentes, a variável escolhida é aquela que produz a menor medida de variação do limite dual. O critério empregado para o cálculo desta medida não é divulgado na documentação disponível do `XPRESS-MP`.

- Após realizarmos testes com as instâncias do PARG de *pequena* dimensão, adotamos os valores MINIMPR = 0.01% e MAXITER = 15.

- Impusemos um tempo máximo de CPU de 4 horas para a o algoritmo Branch-and-Cut. Este limite não leva em conta o tempo gasto pelo algoritmo de pré-processamento, ou seja pelo algoritmo NDRC.

5.3 Experimentos Computacionais

Para testar o algoritmo híbrido proposto neste Capítulo, utilizamos os conjuntos de instâncias R, D-E e D-R descritos no Capítulo 4. Para cada conjunto, testamos o algoritmo Branch-and-Cut para aquelas instâncias onde o algoritmo NDRC sozinho não foi capaz de provar otimalidade.

Foram considerados nos testes quatro variantes de algoritmos Branch-and-Cut, denominadas respectivamente por BC1, BC2, BC3, BC4. Para cada variante, o problema de entrada corresponde ao grafo pré-processado, obtido ao final do algoritmo NDRC. As variantes diferem entre si pelas classes de desigualdades que separam e pelo uso ou não das desigualdades obtidas pelo procedimento de warm-start. A Tabela 5.1 ilustra as diferenças entre as variantes testadas.

Tabela 5.1: Variantes testadas de algoritmos Branch-and-Cut

	BC1	BC2	BC3	BC4
Pré-processamento NDRC	SIM	SIM	SIM	SIM
Separação de SECs e cutsets	SIM	SIM	SIM	SIM
Separação heurística de desigualdades Blossom	NÃO	NÃO	SIM	SIM
Uso das desigualdades SECs provenientes do warm-start	NÃO	SIM	NÃO	SIM
Uso das desigualdades Blossom provenientes do warm-start	NÃO	NÃO	NÃO	SIM

Na primeira variante, BC1, apenas cutsets e SECs são separadas. Nesta variante, não empregamos nenhuma desigualdade (SEC ou Blossom) que tenha sido identifi-

cada pelo procedimento de warm-start descrito na Seção 4.3. Já BC2 emprega, além dos procedimentos de separação utilizados em BC1, as desigualdades SECs obtidas através do procedimento de warm-start da Seção 4.3. No algoritmo BC3, além da separação de cutsets e SECs, separamos também desigualdades Blossom, mas não empregamos nenhuma desigualdade SEC ou Blossom obtida com o procedimento de warm-start da Seção 4.3. Finalmente, no algoritmo BC4, separamos cutsets, SECs e desigualdades Blossom e introduzimos todas as desigualdades SECs e Blossom identificadas pelo warm-start. Cabe salientar que a inicialização do cut-pool para as variantes BC2 e BC4, feita com as desigualdades identificadas pelo warm-start, não representa uma sobrecarrega de memória, ou de tempo para verificar violação, ao longo do algoritmo. Por um lado, o número de desigualdades SECs que identificamos no warm-start é limitado por $O(|V|)$, ou seja o número de iterações do algoritmo guloso. Por outro lado, ao longo das iterações do algortimo NDRC, observamos que, na média, o número de desigualdades Blossom mantidas dualizadas variava em torno de 40% a 60% de $|V|$. Assim sendo, o número de desigualdades introduzidas no cut-pool é da ordem de $O(|V|)$.

Os algoritmos aqui descritos foram implementados em C. Utilizamos o compilador gcc (com chave de otimização -O3 ativada) no sistema operacional Linux. Todos os resultados divulgados nesta Seção foram obtidos em uma máquina com processador AMD ATHLON 2000 com 1Gbyte de memória RAM.

Antes de procedermos a uma comparação direta dos algoritmos aqui testados, é importante relembrar que todos eles utilizam o mesmo grafo de entrada, pré-processado pelo algoritmo NDRC, e os mesmos limites superiores iniciais, também obtidos pelo algoritmo NDRC. Assim sendo, embora BC1 e BC2 não se utilizem diretamente de desigualdades Blossom, acabam por se beneficiar, indiretamente, das mesmas. Isso se aplica pois todas as variantes testadas do algoritmo Branch-and-Cut se beneficiam dos melhores limites inferiores e da fixação de variáveis obtidos com o uso de desigualdades Blossom, na fase NDRC. Portanto, pode-se dizer que uma comparação de BC3 e BC4 com BC1 e BC2 deve levar em conta que BC1 e BC2 receberam uma *ajuda indevida.*

Na Tabela 5.2, para cada variante do algoritmo Branch-and-Cut, apresentamos a média do número de nós e do tempo de computação requerido para se resolver as 12 instâncias do grupo R (não resolvidas pelo algoritmo NDRC sozinho). Comparando os resultados de BC2 com BC1, podemos observar que o uso das desigualdades SECs

propostas na Seção 4.3 reduziu o tempo médio de computação em mais de 4 vezes e o número de nós abertos em cerca de 30%. Para algumas dessas instâncias, BC2 foi cerca de 10 vezes mais rápido que BC1. Entretanto, o uso das desigualdades Blossom, identificadas pelo warm-start ou pela heurística de separação, não melhorou os resultados da variante BC2. Note também que não há diferença apreciável de performance entre BC4 e BC2 ou entre BC3 e BC1. Note ainda que o ganho de desempenho de BC4 sobre BC3 é muito próximo do ganho de desempenho de BC2 sobre a BC1. Isto se deve ao fato do uso das desigualdades Blossom não ter tido impacto apreciável na melhoria de desempenho das instâncias deste grupo de problemas.

Tabela 5.2: Comparação das variantes de algoritmos Branch-and-Cut: média para instâncias do grupo R

	BC1		BC2		BC3		BC4	
	Nós	t(s)	Nós	t(s)	Nós	t(s)	Nós	t(s)
Médias	10.67	67.75	6.50	15.18	10.58	67.96	7.00	15.58

Nas Tabelas 5.3 e 5.4, apresentamos uma comparação direta dos algoritmos para as instâncias D-E e D-R, respectivamente. Para cada algoritmo (em cada Tabela) apresentamos o número de nós abertos na árvore de enumeração e o tempo de CPU (em segundos) gasto na fase Branch-and-Cut. Uma indicação (-) nas colunas de número de nós e de tempo indica que o algoritmo não resolveu a instância no tempo máximo estabelecido.

Analisando os resultados obtidos na Tabela 5.3, para o conjunto D-E, verificamos que os algoritmos que utilizam desigualdades Blossom, ou seja BC3 e BC4, resolveram todas as instâncias testadas, enquanto os algoritmos baseados apenas em SECs e cutsets (BC1 e BC2) não resolveram 3 das 16 instâncias testadas, no tempo limite estabelecido. Considerando a média dos resultados para as 13 instâncias que todos os quatro algoritmos foram capazes de resolver (veja a última linha da Tabela), observamos que BC4 foi cerca de 3.6 vezes mais rápido que BC1. Para isto, colaboraram tanto a separação das desigualdades Blossom (BC3 é cerca de 2 vezes mais rápido que BC1) como a inicialização do cut-pool através do procedimento de

warm-start (BC4 é cerca de 1.8 vezes mais rápido que BC3). Note ainda que, na média, o número de nós necessários para resolver as instâncias cai consideravelmente quando separamos desigualdades Blossom (cerca de 5 vezes) embora se reduza em uma menor proporção (cerca de 10%) sob o efeito do warm-start.

Tabela 5.3: Comparação das variantes de algoritmos Branch-and-Cut: instâncias D-E

	BC1		BC2		BC3		BC4	
Instância	Nós	t(s)	Nós	t(s)	Nós	t(s)	Nós	t(s)
de_100_3	23	1.32	21	0.22	7	1.27	5	0.23
de_200_1	3	2.99	3	0.51	3	2.96	3	0.48
de_200_2	400	53.51	1245	66.11	523	180.95	361	89.26
de_200_3	267	56.50	149	17.79	87	45.01	37	8.64
de_300_1	327	310.10	200	38.60	49	169.06	69	44.70
de_300_2	9	3.16	9	2.26	5	2.47	5	1.18
de_300_3	4523	1985.72	4373	1283.55	311	621.97	65	157.69
de_400_1	117	871.87	73	227.28	17	674.39	1	124.79
de_400_2	480	1381.70	417	1180.12	298	740.10	279	652.21
de_400_3	147	553.98	221	104.88	39	1327.29	43	53.24
de_500_1	295	1862.88	12	574.07	8	140.64	4	48.07
de_500_2	387	3100.18	52	1963.20	17	733.43	10	736.76
de_500_3	-	-	-	-	724	11771.56	626	7074.36
de_600_1	-	-	-	-	626	10619.91	319	5429.24
de_600_2	-	-	-	-	959	13806.47	419	10294.57
de_600_3	145	4532.97	138	3800.43	27	2839.82	9	2132.17
Média	547.92	1132.06	531.76	712.23	107.00	575.34	89.10	311.49

Tabela 5.4: Comparação das variantes de algoritmos Branch-and-Cut: instâncias D-R

	BC1		BC2		BC3		BC4	
Instância	Nós	t(s)	Nós	t(s)	Nós	t(s)	Nós	t(s)
dr_100_1	27	0.48	13	0.18	27	0.49	23	0.27
dr_200_1	39	19.61	23	3.79	39	19.75	31	5.97
dr_200_2	27	10.96	13	0.93	27	10.85	7	1.07
dr_200_3	25	11.45	29	9.29	25	11.42	25	4.97
dr_300_1	41	119.17	27	63.56	31	115.33	11	12.39
dr_300_3	91	220.82	49	59.87	111	280.26	143	161.49
dr_400_1	563	3047.50	151	840.70	200	1297.26	105	543.55
dr_400_2	199	887.77	195	736.94	101	459.95	49	149.92
dr_500_1	127	2454.20	81	802.38	123	2455.10	47	328.20
dr_500_2	65	1150.37	19	224.11	42	893.45	39	695.54
dr_500_3	59	1265.14	89	954.56	59	1264.13	37	317.72
dr_600_1	117	3968.94	63	1493.99	117	3941.40	27	658.72
dr_600_2	181	8195.37	223	7593.87	177	8189.73	97	3683.00
dr_600_3	57	2250.00	153	2006.26	53	2251.43	367	1991.45
Média	115.87	1685.84	80.57	1056.37	80.85	1513.61	78.14	874.08

Os resultados descritos na Tabela 5.4 indicam que todos os quatro algoritmos foram capazes de resolver as 14 instâncias do conjunto D-R. Esses resultados indicam ainda que BC4 foi 48% mais rápido que BC1 e que a separação de desigualdades Blossom proporcionou uma redução de tempo de cerca de 11% (BC3 gastou, na média, 11% menos tempo que BC1). Note também que o uso das desigualdades SECs oriundas do warm-start colaborou decisivamente para um melhor desempenho do algoritmo, se comparamos BC2 com BC1 e BC4 com BC3. Vale ainda ressaltar que, apesar da utilização da heurística de separação de desigualdades Blossom ter colaborado positivamente para a redução dos tempos médios de CPU, o uso das

desigualdades SECs oriundas do warm-start foi o aspecto preponderante para a melhor performance de BC4.

Nas colunas da Tabela 5.5 (conjunto D-E) e da Tabela 5.6 (conjunto D-R) apresentamos, respectivamente, a identificação das instâncias, o custo das soluções ótimas associadas, os tempos gastos na fase NDRC, os tempos de CPU gastos pelo algoritmo BC4 e, finalmente os tempos totais de CPU. Os tempos de CPU relativos ao algoritmo NDRC apresentados nesta Tabela diferem daqueles apresentados no Capítulo 4 já que foram obtidos em computadores diferentes.

Tabela 5.5: Resultados consolidados BC4: instâncias D-E

		t(s)		
Instância	z	NDRC	BC4	TOTAL
de_100_1	8779	6.78	-	6.78
de_100_2	9629	5.20	-	5.20
de_100_3	10798	17.89	0.23	18.12
de_200_1	12031	76.77	0.48	77.25
de_200_2	12645	140.70	89.26	229.96
de_200_3	12229	110.91	8.64	119.55
de_300_1	14792	318.69	44.70	362.70
de_300_2	13931	128.45	1.18	129.63
de_300_3	14997	325.20	157.69	482.89
de_400_1	19239	1167.69	124.79	1292.48
de_400_2	17419	1090.75	652.21	1742.96
de_400_3	16091	537.43	53.24	590.67
de_500_1	19357	1181.08	48.07	1229.48
de_500_2	22400	3438.65	736.76	4175.41
de_500_3	19411	2074.56	7074.36	9148.92
de_600_1	21930	3245.30	5429.24	8674.54
de_600_2	21449	3374.61	10294.57	13669.18
de_600_3	22243	4556.33	2132.17	6688.50

Tabela 5.6: Resultados consolidados BC4: instâncias D-R

Instância	z	t(s) NDRC	BC4	TOTAL
dr_100_1	2186	8.77	0.27	9.04
dr_100_2	2674	7.39	-	7.39
dr_100_3	2689	10.41	-	10.41
dr_200_1	2141	67.76	5.97	73.73
dr_200_2	2471	89.36	1.07	90.43
dr_200_3	2405	74.87	4.97	79.84
dr_300_1	2462	230.96	12.39	243.35
dr_300_2	2312	155.19	-	155.19
dr_300_3	2585	333.99	161.49	495.48
dr_400_1	2817	1215.58	543.55	1759.85
dr_400_2	2443	720.94	149.92	870.87
dr_400_3	2387	269.08	-	269.08
dr_500_1	2597	1033.65	328.20	1361.85
dr_500_2	2652	3914.84	695.54	4610.38
dr_500_3	2593	1207.32	317.72	1525.04
dr_600_1	2820	2577.32	658.72	3236.04
dr_600_2	2662	3112.32	3683.00	6795.32
dr_600_3	2800	2293.15	1991.45	4284.60

Comparando os resultados das Tabelas 5.5 e 5.6 com aqueles das Tabelas 4.5 e 4.6, observamos que, na média, a diferença entre os limites inferiores obtidos pelo algoritmo NDRC e os valores associados às soluções ótimas correspondentes são, para as instâncias D-E e D-R de 0.18% e 0.08%, respectivamente. Já as diferenças médias observadas entre limites superiores e valores de soluções ótimas são de 0.68% e 1.12%, respectivamente. Note que, por um lado, os limites inferiores obtidos pelo algoritmo NDRC para as instâncias D-R estão mais próximos do ótimo que aqueles obtidos para as instâncias D-E. No entanto, por outro lado, os limites superiores obtidos pelo algoritmo NDRC para o conjunto D-R são piores que aqueles obtidos para o conjunto D-E. À luz destes resultados, não há uma indicação clara de que as

instâncias randômicas sejam mais difíceis de se resolver que instâncias Euclidianas. Por um lado, se temos à disposição soluções ótimas para uma instância D-E e para uma instância D-R, de iguais dimensões, parece-nos razoável acreditar que, em função dos resultados obtidos, provar a otimalidade (através de limites duais) é mais difícil para a instância D-E que para a instância D-R. Por outro lado, as heurísticas primais que propusemos parecem funcionar melhor diante de custos Euclidianos.

5.4 Comentários finais

Em linhas gerais, considerando a experiência computacional descrita neste Capítulo e no anterior, podemos dizer que as desigualdades Blossom tiveram um impacto bastante positivo, principalmente para a solução ótima de instâncias Euclidianas, com limites superiores de grau muito justos. Para as demais instâncias o emprego das desigualdades Blossom contribuiu de forma menos significativa. Contudo, as variantes de algoritmos Branch-and-Cut que usam tais desigualdades (BC3, BC4) não apresentaram (salvo um único caso isolado, para a instância de_200_2) performance inferior às variantes baseadas apenas na formulação SEC (BC1, BC2). Observamos também que, independentemente do tipo de instância testada, o procedimento de warm-start proposto na Seção 4.3 contribuiu decisivamente para a redução dos tempos de CPU associados ao algoritmo Branch-and-Cut. Assim sendo, para qualquer conjunto de instâncias aqui considerado, acreditamos que o nosso algoritmo híbrido domine (em termos de tempo de CPU) o único algoritmo Branch-and-Cut para o PARG disponível na literatura [15].

Os resultados deste Capítulo confirmam a tese de [1] que contradiz a visão previamente estabelecida na literatura de que instâncias do PARG com custos randômicos são mais fáceis de resolver que aquelas com custos Euclidianos. Da mesma forma, como pôde ser apreciado neste trabalho, confirmamos a tese de que o limite de grau dos vértices pode ser um parâmetro determinante na dificuldade de resolução de uma instância do PARG.

Em uma etapa posterior a este trabalho, pretendemos utilizar o método de geração de colunas (em conjunto com a geração de cortes) para viabilizar a solução de instâncias do PARG com dimensões superiores àquelas aqui tratadas. Assim procedendo, os problemas de programação linear a se considerar irão envolver um pequeno número de variáveis, ou seja, apenas aquelas variáveis com custos reduzidos

não negativos.

Encerramos este Capítulo com uma breve discussão sobre a possível utilização de desigualdades Combs e Clique Trees, futuramente. A separação de desigualdades Combs em relação ao politopo do $PCV(n)$ tem sido bastante estudada. Um dos resultados teóricos recentes mais importantes neste sentido é devido a Letchford e Lodi [83]. Neste trabalho, os autores estudaram a sepação exata de uma classe de desigualdades Combs, as *simple combs*.

Dizemos que uma desigualdade Comb é *simples* se: ou $|T_i \cap H| = 1$ ou $|T_i \setminus H| = 1$ para todo $i = 1, ..., k$, ou se as duas condições anteriores são simultaneamente satisfeitas. Baseando-se na equivalência poliedral (no caso do $PCV(n)$) entre as desigualdades cutsets ($x(\delta(W)) \geq 2$, $\forall W \in V$) e as SECs (3.3), Letchford e Lodi lançaram mão de um resultado de Caprara e Fischetti [19] (veja o Teorema 4 em [19] e o Teorema 2 em [83]) e apresentaram um algoritmo exato, de complexidade $O(|V|^3|E|^3\log(|V|)$ para a separação de Combs simples. Este algoritmo é válido caso o ponto a ser separado satisfaça a todas as desigualdades de eliminação de subrotas (3.3).

Diferentemente do $PCV(n)$, no caso do $PARG(n)$, a satisfação das desigualdades cutsets (3.6) não implica na satisfação das SECs (3.3). Na ausência desta hipótese, não é válido utilizar em nosso contexto, o mesmo resultado de [19] utilizado em [83] para a proposição do algoritmo de separação de Combs, ali descrito.

Cabe salientar que, ainda que esta dificuldade tivesse sido vencida, o uso deste algoritmo, na forma como sugerido em [83], seria provavelmente impraticável em um algoritmo Branch-and-Cut. Isto porque o ponto a ser separado necessitaria satisfazer a todas as desigualdades SECs e, em função do efeito do tailing off, esta condição poderia levar a um gasto excessivo de CPU, em cada nó da árvore de enumeração (principalmente nos primeiros). Por outro lado, se o efeito do tailing off não fosse uma consideração real, na prática, a elevada complexidade do algoritmo exato de [83] talvez o limitasse em um algoritmo Branch-and-Cut competitivo.

As mesmas dificuldades apontadas nos parágrafos anteriores nos impediram de generalizar para o PARG o algoritmo proposto por Carr [20] para a separação, no politopo $PCV(n)$, de uma classe especial de desigualdades Clique-Trees.

Em trabalhos futuros, pretendemos sugerir e testar heurísticas para a separação de desigualdades Combs para o PARG, baseadas, eventualmente, na adaptação de heurísticas de separação de Combs para o $PCV(n)$ (veja [100, 101] para detalhes).

Alguns destes procedimentos baseiam-se na identificação de conjuntos *handles e teeth* candidatos que, se combinados entre si, podem resultar em desigualdades Combs violadas. No caso do PCV, a identificação destes conjuntos candidatos é facilitada pela hipótese de matching perfeito. Dado que desigualdades Combs são uma generalização das desigualdades Blossom, acreditamos que as instâncias do conjunto D-E poderão se beneficiar positivamente do uso das desigualdades Combs, uma vez que, para esse conjunto, as Relaxações Lineares baseadas nas desigualdades Blossom se mostrou mais vantajosa que aquela baseada unicamente em SECs.

Capítulo 6

Uma heurística Lagrangeana para o Problema de Steiner em Grafos com Recolha de Prêmios

6.1 Introdução

Seja $G = (V, E)$ um grafo não direcionado com conjunto de vértices V e conjunto de arestas E. Assuma que penalidades $\{d_i \geq 0 : i \in V\}$ e custos $\{c_e \geq 0 : e \in E\}$ sejam associados aos vértices e arestas de G, respectivamente. Definimos uma Árvore de Steiner com Recolha de Prêmios (ASRP) de G como um vértice isolado ou, caso contrário, como uma árvore qualquer de G. A cada ASRP associamos um peso, dado pela soma dos custos de suas arestas mais a soma das penalidades dos vértices por ela não cobertos. O Problema de Steiner em Grafos com Recolha de Prêmios (PASRP) consiste então em obter uma ASRP de mínimo peso.

O PASRP generaliza um problema bastante estudado em Otimização Combinatória, o Problema de Steiner em Grafos (PSG) [93, 124]. Neste, um conjunto de vértices ditos *terminais* deve ser conectado por qualquer árvore viável. Possivelmente, para conectá-los, pode-se eventualmente empregar alguns outros vértices não terminais de V. Os vértices não terminais presentes em uma Árvore de Steiner são denominados de *vértices de Steiner*. Não é difícil perceber que o PSG pode ser visto como um PASRP no qual penalidades *suficientemente grandes* são associadas aos vértices terminais e penalidades nulas são associadas aos demais vértices de V. Quando essas penalidades são impostas, uma ASRP ótima deve necessariamente conectar todos os vértices terminais como exigido no PSG. Assim sendo, como o PSG é $\mathcal{NP}$-completo [71], a versão de decisão do PASRP também o é.

Recentemente, o PASRP tem sido foco de considerável atenção na literatura.

Uma possível explicação para isso está relacionada à capacidade do problema em retratar apropriadamente uma série de aplicações em *projeto de redes*. Um dos primeiros exemplos nesse sentido é relatado em [92] e diz respeito à construção de uma rede de fibra ótica para prover serviços de conexão de banda larga a clientes residenciais e comerciais. Neste caso, o grafo G representa o mapa da região para a qual está sendo projetada: V denota o conjunto de potenciais clientes e entroncamentos entre ruas e E representa as ruas propriamente ditas. Em outra aplicação, descrita em [85], considera-se o planejamento ótimo da expansão da distribuição de energia elétrica e gás natural. Nesse contexto, as penalidades dos vértices representam o valor presente dos fluxos de caixa associados ao consumo previsto do insumo em cada cliente potencial. Os pesos das arestas, por sua vez, representam o custo de construir (ou eventualmente expandir) redes de tubulações, cabos elétricos, etc.

Pelo que conhecemos, o termo *Recolha de Prêmios* foi introduzido por Balas [4] no contexto do Problema do Caixeiro Viajante. Desde então, tem sido empregado para qualificar problemas onde há uma clara relação de compromisso entre realizar os investimentos necessários para incluir um vértice adicional a uma dada rede (ou solução) ou incorrer em uma penalidade por deixá-lo fora da mesma.

Provavelmente, um dos primeiros algoritmos exatos propostos para resolver problemas similares ao PASRP é aquele intoduzido em Segev [118]. Nesta referência, o Problema da Árvore de Steiner com Peso nos Vértices (PASPV) é usado para descrever uma variante do PSG na qual um vértice raiz deve necessariamente estar presente em qualquer solução. Em adição aos custos associados às arestas, no PASPV, penalidades são também associadas aos demais vértices do grafo de definição do problema.

Baseando-se em uma formulação similar àquela sugerida para o PSG em [87], Engewall et al. [39] apresentaram um algoritmo para o PASPV baseado em Relaxação Lagrangeana. Naquele algoritmo, os limites inferiores para o problema foram gerados de forma similar à metodologia introduzida em Lucena [87] para o PSG. Assim sendo, o algoritmo de [39] pode ser considerado um algoritmo do tipo NDRC para o PASPV.

O PASRP, na forma de minimização descrita aqui, pode ser polinomialmente aproximado [31, 54]. O primeiro trabalho que conhecemos abordando esse aspecto foi proposto por Biesntock [31]. Naquela referência, é descrito um algoritmo de aproximação de fator 3 para o PASRP. Posteriormente, Goemans e Williamson [54]

propuseram uma metodologia geral (do tipo primal-dual) para a aproximação de uma família de problemas $\mathcal{NP}-$ completos que inclui o PASRP. A especialização para o PASRP do algoritmo em [54] leva a um fator de aproximação 2 e tem complexidade $O(n^2\log(n))$ ($n = |V|$). Originalmente, esse algoritmo foi sugerido para uma versão roteada do PASRP, na qual um vértice raiz necessariamente deve pertencer a qualquer solução viável do problema. Para o caso geral, onde um vértice raiz não é imposto, o algoritmo deve ser executado $|V|$ vezes; uma vez para cada *raiz* $r \in V$ diferente de V. Assim procedendo, o fator de aproximação 2 é preservado para o caso geral, enquanto a complexidade envolvida é elevada para $O(n^3\log(n))$. Recentemente, Minkoff [99] aprimorou o algoritmo de Goemans and Williamson (GW) ao abolir a exigência da existência de um vértice raiz para a aplicação do mesmo. O algoritmo descrito em [99] preservou o fator de aproximação assintótico do algoritmo GW e reduziu sua complexidade para $O(n^2\log(n))$. Uma contribuição adicional daquele estudo consiste no desenvolvimento de estratégias de pós-processamento (ou *pruning*), baseados em Programação Dinâmica, que dominam aquelas propostas no algoritmo GW.

Ao especializar a formulação do PSG [53, 87, 95] para o PASRP, Lucena e Resende [92] introduziram a primeira formulação específica para o problema. Associado àquela formulação, Lucena e Resende apresentaram um algoritmo de Planos de Corte baseado na separação exata de *desigualdades de eliminação de subrotas generalizadas* (GSECs). Os resultados computacionais obtidos indicaram que a formulação sugerida em [92] é bastante forte, uma vez que levou a soluções inteiras para praticamente todas as instâncias então existentes na literatura. Adicionalmente, testes de redução bastante eficientes, adaptados da literatura associada ao PSG [33], foram também propostos em [92].

Outra formulação de Programação Inteira para o PASRP foi introduzida por Ljubíc et al. [85]. A formulação baseia-se em um digrafo induzido por G e envolve um vértice raiz. Os resultados computacionais obtidos através de um algoritmo Branch-and-Cut baseado naquela formulação foram expressivos. Além de resolver todas as instâncias até então disponíveis na literatura, o algoritmo resolveu também, com garantia de otimalidade e sob tempos de CPU bastante razoáveis, muitas das novas (e mais difíceis) instâncias introduzidas no estudo. Além disto, para muitas instâncias, o bom desempenho do algoritmo foi pouco ou nada dependente dos testes de redução propostos para o PASRP.

Recentemente, Haouari e Siala [66] estudaram o PASRP *sob cotas*. Nesta variante do problema, não são impostas penalidades aos vértices de G. Ao invés disto, um *prêmio* é recolhido sempre que um vértice é conectado à uma árvore. Para que seja viável, a soma dos prêmios associados aos vértices cobertos pela árvore deve ser igual ou superior a um valor mínimo pré estabelecido. Na prática, estas modificações implicam na introdução de uma desigualdade da mochila na formulação do problema. Assim sendo, a definição do PASRP sob cotas é mais próxima da idéia de *recolha de prêmios*, originalmente introduzida por Balas [4]. Para resolver o problema, Haouari e Siala propuseram um método de solução híbrido baseado em Relaxação Lagrangeana e Algoritmos Genéticos. A metodologia foi testada em instâncias envolvendo até 500 vértices e 5000 arestas.

Em outra linha de pesquisa, Canuto et al. [18] propuseram uma Metaheurística para resolver o problema. O procedimento combina vários ingredientes: uma heurística construtiva do tipo *multi-start* (baseada no algoritmo de Goemans e Williamson com custos de entrada aleatoriamente perturbados), um procedimento de Busca Local (BL) e de Busca em Vizinhança Variável (VNS) e, finalmente, um procedimento de Path-Relinking (PR). Os inúmeros experimentos computacionais conduzidos naquele estudo demonstraram que os métodos de solução ali propostos foram capazes de obter soluções ótimas ou *quase-ótimas* em tempos de computação bastante razoáveis para muitas das instâncias então disponíveis para o problema.

Neste Capítulo, descrevemos uma heurística Lagrangeana para o PASRP. No esquema aqui proposto, o método de Relaxação Lagrangeana é aplicado à formulação do PASRP introduzida em [92]. À primeira vista, esta formulação não parece adequada à utilização de Relaxação Lagrangeana. Procedemos então a uma reformulação da mesma, introduzindo um novo conjunto de variáveis que confere ao PASRP uma interpretação em grafos mais conveniente. Ao fazer isto, estamos seguindo idéias anteriormente introduzidas em [89, 90] para reformular e tratar algoritmicamente a formulação do PSG introduzida em [53, 86, 95]. Sob essa reformulação, o PASRP torna-se bastante atraente para ser decomposto de forma Lagrangeana e um algoritmo NDRC será assim descrito para o problema. Este algoritmo emprega informação dual Lagrangeana para direcionar a geração de soluções viáveis para o PASRP. Essa heurística, que envolve uma fase construtiva e uma Busca Local, é chamada repetidas vezes ao longo da execução do algoritmo NDRC. Neste, por sua vez, testes de pré-processamento e de fixação de variáveis são também aplicados. Final-

mente, concluindo o estudo, implementamos um algoritmo Branch-and-Cut baseado na formulação do PASRP introduzida em [92] e o comparamos com a heurística Lagrangeana aqui sugerida.

Este Capítulo é organizado da seguinte forma. Na Seção 6.2 apresentamos a formulação para o PASRP introduzida em [92]. Na Seção 6.2.1, uma reformulação é proposta para permitir que um algoritmo NDRC seja utilizado para resolver o problema. Na Seção 6.3, descrevemos os procedimentos empregados para obter limites Lagrangeanos para o PASRP. Testes de pré processamento e de fixação de variáveis para o problema são descritos na Seção 6.4. Resultados computacionais obtidos pela heurística Lagrangeana são apresentados na Seção 6.5. Na Seção 6.6, descrevemos o algoritmo Branch-and-Cut que implementamos para o PASRP. Finalmente, encerramos o Capítulo na Seção 6.7 com algumas considerações finais.

6.2 Uma formulação de Programação Inteira para o PASRP

Nesta Seção, inicialmente apresentamos a formulação de Programação Inteira para o PASRP introduzida em [92]. Posteriormente, procedemos à sua reformulação.

A formulação sugerida para o PASRP em [92] envolve dois conjuntos distintos de variáveis: $\{y_i \in \{0,1\} : i \in V\}$ para selecionar os vértices de uma ASRP e $\{x_e \geq 0 : e \in E\}$ para escolher as arestas a serem empregadas para conectar os vértices da mesma. Como nos Capítulos anteriores, denotamos por $E(S) \subseteq E$ o conjunto de arestas com ambas as extremidades em $S \subseteq V$, por $x(E(S)) := \sum_{e \in E(S)} x_e$ a soma das variáveis de decisão associadas às arestas em $E(S)$ e, similarmente, por $y(S) := \sum_{i \in S} y_i$ a soma das variáveis de decisão associadas aos vértices de S. A formulação se deriva daquela proposta para o PSG em [53, 87, 95] e é dada por:

$$w = \min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R}_0 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|}) \right\}, \tag{6.1}$$

onde a região poliedral $\mathcal{R}_0$ é definida como:

$$x(E) = y(V) - 1, \tag{6.2}$$

$$x(E(S)) \leq y(S \setminus \{j\}), \forall j \in S, \forall S \subseteq V, \tag{6.3}$$

$$0 \leq x_e \leq 1, \forall e \in E, \tag{6.4}$$

$$0 \leq y_i \leq 1, \forall i \in V. \tag{6.5}$$

Na formulação acima, as restrições (6.2) impõem que o número de arestas selecionadas em uma árvore viável seja exatamente igual ao número de vértices cobertos pela mesma menos uma unidade. As restrições (6.3) são GSECs e garantem que a solução resultante é livre de ciclos. Note que se $y_i = 1$, para todo $i \in S$, as restrições (6.3) se reduzem as SECs (3.3). O conjunto de soluções inteiras dados por (6.1) correspondem então às ASRP, conforme previamente definidas.

Observe que as soluções do PASRP correspondentes a vértices isolados podem ser eficientemente determinadas por enumeração direta. Baseando-se nisto, Lucena e Resende [92] concentraram-se nas soluções do problema que envolvessem uma ou mais arestas. Uma formulação restrita do problema, excluindo vértices isolados, é então dada por:

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R}_1 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|}) \right\}, \tag{6.6}$$

onde a região poliedral $\mathcal{R}_1$ é definda pelo conjunto de restrições em $\mathcal{R}_0$ e pelas seguintes desigualdades:

$$x(\delta(i)) \geq y_i, \forall i \in V \text{ para } d_i > 0, \tag{6.7}$$

$$x(\delta(i)) \geq 2y_i, \forall i \in V \text{ para } d_i = 0. \tag{6.8}$$

Note que ao impor que soluções para o PASRP devem possuir pelo menos uma aresta de G, as desigualdades (6.7) e (6.8) se beneficiam do fato de que os custos associados às arestas e as penalidades associadas aos vértices em (6.6) são não negativas. Assim sendo, as desigualdades (6.7) impõem que deve haver pelo menos uma aresta incidente a um vértice com penalidade positiva em uma ASRP. Por outro lado, para os vértices com penalidades nulas, as desigualdades (6.8) impõem que tais vértices não podem ser folhas de uma solução ótima. Isso resulta do fato de que a exclusão dessas folhas resultaria em uma solução viável de menor custo para o PASRP. Nesse caso, como resultado, a otimalidade da solução original estaria sendo contradita.

6.2.1 Uma estrutura adequada para decomposição Lagrangeana

Como sugerido anteriormente, ao aplicarmos Relaxação Lagrangeana a um PI, procuramos identificar um conjunto de restrições que, se relaxado, torna o problema

remanescente fácil de ser resolvido. Em princípio, a região poliedral $\mathcal{R}_1$ não parece conter uma estrutura adequada a estes propósitos. Contudo, como veremos a seguir, essa estrutura desejada encontra-se *oculta* em $\mathcal{R}_1$ e será *revelada* por um procedimento baseado em dois passos. No primeiro deles, substituiremos as variáveis binárias $\{y_i : i \in V\}$ por seus complementos em 1. Em seguida, o novo conjunto de variáveis será reinterpretado em termos de um novo grafo, que expande $G = (V, E)$ com a introdução de um vértice artificial, em conjunto com algumas arestas a ele incidentes.

Para implementar o primeiro dos dois passos sugeridos acima, seja $\{z_i = 1 - y_i : i \in V\}$ o conjunto de variáveis que substituirá $\{y_i : i \in V\}$ em (6.6). Esta substituição, que resulta em uma nova região poliedral $\mathcal{R}_2$ (em uma correspondência direta com $\mathcal{R}_1$) é dada por:

$$x(E) + z(V) = |V| - 1, \tag{6.9}$$

$$x(\delta(i)) + z_i \geq 1, \forall i \in V \text{ para } d_i > 0, \tag{6.10}$$

$$x(\delta(i)) + 2z_i \geq 2, \forall i \in V \text{ para } d_i = 0, \tag{6.11}$$

$$x(E(S)) + z(S \setminus \{j\}) \leq |S| - 1, \forall j \in S, \forall S \subseteq V, \tag{6.12}$$

$$0 \leq x_e \leq 1, \forall e \in E, \tag{6.13}$$

$$0 \leq z_i \leq 1, \forall i \in V. \tag{6.14}$$

Uma reformulação de (6.6) é então dada por

$$\min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v z_v : (x, z) \in \mathcal{R}_2 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|}) \right\}. \tag{6.15}$$

Note que as GSECs na forma como se apresentam em (6.12) aparecem claramente como um *lifting* das tradicionais SECs (3.3).

Antes de prosseguirmos, vamos primeiro sugerir uma interpretação mais adequada das variáveis $\{z_i : i \in V\}$. Para tanto, assuma que um vértice artificial $n+1$, onde $n = |V|$, tenha sido introduzido em $G = (V, E)$. Assuma ainda que toda variável z_i, $i \in V$ represente uma aresta de custo d_i que liga o vértice $i \in V$ ao vértice $n+1$. Denotando por $G' = (V', E')$ o grafo que resulta desta expansão de G, temos que $V' = V \cup \{n+1\}$ e $E' = E \cup \{(i, n+1) : i \in V\}$.

Suponha que $V_c \subseteq V$, tal que $|V_c| \geq 2$, denote o conjunto de vértices que são conectados por uma ASRP que satisfaz (6.9)–(6.14). No grafo G', tal árvore induz

$|V \setminus V_c|$ arestas em $E' \setminus E$ e $|V_c|-1$ arestas em E. Logo, $|V'|-2$ (ou, equivalentemente, $|V|-1$) arestas de G' são induzidas por qualquer solução viável de (6.9)–(6.14). Note também que nenhuma destas soluções pode violar alguma GSEC. Não é difícil verificar que qualquer solução para (6.9)–(6.14) corresponde a uma *floresta restrita* de G' com exatamente $|V'|-2$ arestas e duas componentes conexas. Uma destas componentes conexas deve ser uma *estrela* que tenha $n+1$ como vértice central, ou seja, um conjunto de uma ou mais arestas de G', todas incidentes ao vértice $n+1$, ou, alternativamente, essa componente pode ser o próprio vértice $n+1$ isolado. Observe que as GSECs de cardinalidade 2, como por exemplo $x_e + z_i \leq 1$ e $x_e + z_j \leq 1$, para $S = \{i, j\} \subset V$ e $e = (i, j) \in E$ garantem a topologia dessa componente. A outra componente conexa da floresta restrita deve ser uma ASRP envolvendo pelo menos uma aresta de G, satisfazendo as restrições de grau (6.10) e (6.11). Uma ASRP típica, diante da interpretação dada ao grafo expandido G', é indicada na Figura 6.1.

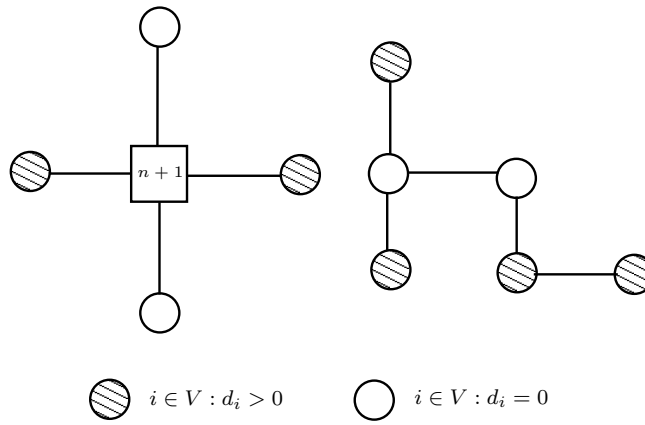

Figura 6.1: Uma ASRP no grafo expandido G'

Observe que, sob a interpretação proposta acima, a região $\mathcal{R}_2$ nos oferece uma estrutura bastante interessante para se explorar no contexto de decomposição Lagrangeana: uma floresta irrestrita de G' com exatas $|V'|-2$ arestas. Uma vez que não viole as restrições de grau (6.10)–(6.11) e as GSECs (6.12), tal floresta deve, necessariamente, induzir uma ASRP viável.

Considere assim, a seguinte região poliedral $\mathcal{R}_3$ dada por

$$x(E) + z(V) = |V'| - 2, \tag{6.16}$$

$$x(E(S)) \leq |S| - 1, \forall S \subseteq V', \tag{6.17}$$

$$0 \leq x_e \leq 1, \forall e \in E, \tag{6.18}$$

$$0 \leq z_i \leq 1, \forall i \in V. \tag{6.19}$$

onde (6.17) são as SECs (3.3) já mencionadas. Uma floresta de G' com exatas $|V'|-2$ arestas é associada a qualquer vetor de incidência em $\mathcal{R}_3$. Consequentemente, na medida em que não viola (6.7)–(6.8) e (6.12) este vetor deve induzir uma ASRP viável. Assim sendo, se associarmos multiplicadores de Lagrange não negativos às desigualdades (6.7)–(6.8) e (6.12) e as levarmos à função objetivo em (6.15), a otimização da função Lagrangeana resultante no espaço $(x, z) \in \mathcal{R}_3 \cap (\mathbb{R}_+^{|E|}, \mathbb{B}^{|V|})$ nos fornece um limite inferior válido para o PASRP.

É importante mencionar que Beasley [9] apresentou um procedimento similar ao que acabamos de descrever para a identificação de uma estrutura adequada para o uso de Relaxação Lagrangeana, no contexto do PSG. Similarmente, em [9], introduziu-se no grafo de definição do PSG um vértice artificial e arestas artificiais, todas com custo nulo. Estas arestas artificiais conectam o vértice artificial a alguns vértices do grafo original. Para cada vértice não terminal, uma aresta artificial foi introduzida. Além destas, introduziu-se também uma aresta conectando o vértice artificial a um dos vértices terminais do PSG. Assim sendo, uma solução viável para o problema no grafo resultante desta expansão consiste em uma árvore de custo mínimo, sujeita à restrições adicionais que impõem que os vértices não terminais conectados ao vértice artificial (por meio de suas respectivas arestas artificiais) não podem se conectar (diretamente) a outros vértices do problema. Cabe destacar, entretanto, que na formulação empregada em [9] para o PSG, tais restrições adicionais são impostas apenas pelas GSECs obtidas para $|S| = 2$. As GSECs mais gerais (geradas para $|S| \geq 3$) não foram consideradas em [9], como fazemos aqui através das restrições (6.12). Uma vez que o número destas desigualdades candidatas à dualização é exponencial, recorremos a um procedimento NDRC para obter limites inferiores válidos para o PASRP, sob o Problema Lagrangeano sugerido acima. Este assunto é tratado a seguir.

6.3 Limites Lagrangeanos para o PASRP

6.3.1 Limites inferiores para o PASRP

O algoritmo NDRC a ser aqui descrito para o PASRP é baseado na dualização dinâmica de GSECs (6.12) e na dualizaçao estática das desigualdades de grau (6.10)–(6.11), isto é: algumas das desigualdades (6.12) são dualizadas na medida em que

são violadas, enquanto que as desigualdades (6.10)–(6.11) permanecem dualizadas durante toda a aplicação do Método do Subgradiente (veja a Seção 2.4 para detalhes).

Assuma que, em uma dada iteração k do MS, p Multiplicadores de Lagrange $\lambda^k \in \mathbb{R}^p_+$ sejam associados às desigualdades dualizadas. De acordo com as observações da Seção anterior, um limite inferior válido para w pode ser obtido ao ser resolver o Problema Lagrangeano PL(λ^k) definido como:

$$w_{\lambda^k} = \min \left\{ \sum_{e \in E} c^k_e x_e + \sum_{v \in V} d^k_v z_v + \text{const}(\lambda^k) : (x, z) \in \mathcal{R}_3 \cap (\mathbb{Z}^{|E|}, \mathbb{B}^{|V|}) \right\}, \quad (6.20)$$

onde $\{c^k_e : e \in E\}$ e $\{d^k_i : i \in V\}$ são, respectivamente, os custos e penalidades modificados pelos Multiplicadores de Lagrange e const(λ^k) é uma constante que resulta da dualização das p desigualdades com multiplicadores $\lambda^k \in \mathbb{R}^p_+$.

Observe que uma solução ótima $\overline{x}^k$ para LP(λ^k), isto é, uma floresta de mínimo custo Lagrangeano de G' com exatas $(|V'| - 2)$ arestas, pode ser facilmente encontrada. Para tanto, basta adaptarmos um algortimo que determine uma Árvore Geradora Mínima de G' (por exemplo o algoritmo de Kruskal [77] ou de Prim [111]) para terminar logo após a inserção da $(|V'| - 2)$-ésima aresta.

Concentremo-nos agora na análise das soluções de LP(λ^k) que são inviáveis para o PASRP. Na Figura 6.2, apresentamos o grafo induzido por uma destas soluções. A partir das considerações anteriores, é claro que as soluções de LP(λ^k) que são também válidas para (6.15) devem possuir o vértice $n+1$ aparecendo isoladamente ou, caso contrário, devem ser tais que este vértice é o centro de uma estrela, conforme definido previamente. No caso particular da Figura 6.2, há duas arestas incidentes ao vértice $i \in V$, a saber, $(i, n+1)$ e (i, l). Assim sendo, a solução indicada é inviável para o PASRP e deve envolver desigualdades GSECs violadas.

A identificação de conjuntos que induzem GSECs violadas é um aspecto de crucial importância em nosso algoritmo NDRC. Dada uma soluçao $\overline{x}^k$ para LP(λ^k), a separação destas desigualdades pode ser feita eficientemente. Assuma que $\overline{x}^k$ e o grafo por ela induzido sejam dados. Assuma também que a aresta $(i, n+1)$, para $i \in V$, esteja presente nesta solução. Denote agora por S_i o conjunto de vértices em V (i incluído) que são alcançados a partir de i através de caminhos no grafo induzido por $\overline{x}^k$ que não utilizam a aresta $(i, n+1)$. Sempre que $|S_i| \geq 2$, pelo menos $|S_i| - 1$ desigualdades GSECs violadas (por exatamente uma unidade) devem existir.

Os vértices em S_i, por sua vez, podem ser identificados em tempo proporcional

a $O(n)$. Isto pode ser realizado através da adaptação, para a nossa aplicação, de um algoritmo que determine caminhos mínimos a partir de um dado vértice (no caso, o vértice i) para os demais vértices do grafo (isto é, todos os vértices na mesma componente conexa de i, no grafo induzido por $\overline{x}^k$, após a remoção da aresta $(i, n+1)$).

Observe que qualquer desigualdade GSEC induzida por um subconjunto de S_i ou não é violada, ou é violada em uma unidade. Assim sendo, para cada conjunto S_i, adotamos o critério de, dentre as GSECs violadas, dualizar apenas uma delas. De acordo com este critério, dualizamos aquela desigualdade em (6.12) para a qual $S = S_i$ ($|S_i| \geq 3$) e o vértice j é escolhido como $j \in \arg\max_{l \in (S \setminus i)} \{d_l^k\}$ (empates são resolvidos aleatoriamente).

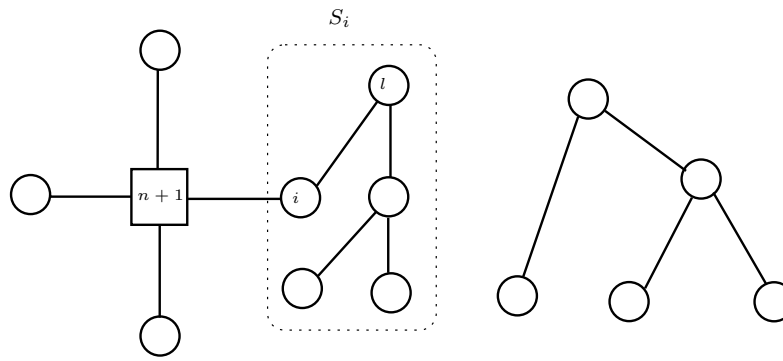

Figura 6.2: Uma solução $\overline{x}^k$ inviável para o PASRP

Tantos quantos forem os vértices $i \in V$ que se conectem, simultaneamente, no grafo induzido por $\overline{x}^k$, ao vértice $n+1$ e a algum outro vértice $l \in V$, tantas serão as desigualdades GSECs dualizadas através do critério acima.

Após a realização de experimentos computacionais, verificamos ser vantajosa a dualização estática de todas as $2|E|$ desigualdades GSECs para as quais $|S| = 2$, em adição às desigualdades (6.10)–(6.11). Estas desigualdades GSECs estão presentes em pequeno número e são as que mais contribuem para o limite dado pelo Problema Dual Lagrangeano

$$w_d = \max_{\lambda \in \mathbb{R}_+^p} \{w_\lambda\}, \tag{6.21}$$

que desejamos obter.

6.3.2 Permitindo a sobrevida de desigualdades dinamicamente dualizadas

Suponha que para o algoritmo NDRC descrito na Seção 6.3.1, uma dada desigualdade tenha sido dualizada dinamicamente na iteração $k \geq 1$ do MS. Suponha também que tal desigualdade seja eliminada do cálculo do passo (2.8) na primeira iteração $k_1 > k$ do MS para a qual, simultaneamente, não seja violada pela solução $\overline{x}^{k_1}$ de LP(λ^{k_1}) e seu multiplicador seja nulo.

Para os experimentos computacionais apresentados neste Capítulo, testamos uma regra alternativa à descrita acima. Precisamente, permitimos uma *sobrevida* da desigualdade no cálculo do passo em (2.8) após a iteração k_1 definida acima. Para melhor entender esta regra, designamos por *idade* de uma desigualdade dinamicamente dualizada, o número de iterações consecutivas do MS após a iteração k_1, durante as quais a desigualdade não é violada pelas soluções dos correspondentes PL(λ). Sob a regra agora introduzida, permitimos o uso de tal desigualdade para o cálculo do passo enquanto sua idade não exceder um valor máximo, denominado aqui de IDADEMAX.

O emprego desta regra alternativa em nosso algoritmo para o PASRP foi bastante efetivo. Como poderá ser constatado na Seção 6.5, para várias instâncias da literatura, conseguimos, através da mesma, obter limites inferiores mais fortes que aqueles obtidos sob a regra de cálculo do passo anteriormente proposta.

6.3.3 Limites superiores para o PASRP

Operando sob um esquema de Relaxação Lagrangeana, os procedimentos que utilizamos para a obtenção de soluções viáveis para o PASRP envolvem duas etapas principais. A primeira é uma heurística construtiva baseada na variante de Minkoff [99] do algoritmo GW [54]. A segunda é uma Busca Local. A seguir, discutimos em detalhes cada uma destas etapas.

6.3.3.1 Uma heurística construtiva *multi-start* para o PASRP

Tanto o algoritmo de GW quanto sua variante proposta por Minkoff [99] (AM) são algoritmos de aproximação primais-duais para o PASRP. Ambos possuem um fator de aproximação assintótico de valor 2 e iniciam-se com uma fase construtiva na qual uma floresta de G é construída sob um critério guloso. À seguir, em ambos os algoritmos, procede-se então a uma etapa de pós processamento ou *pruning*. Nesta,

procura-se encontrar, para os conjuntos de vértices em uma ou mais componentes conexas da floresta obtida na primeira fase do algoritmo, uma ASRP de peso inferior ao peso da componente.

Uma diferença entre os algoritmos GW e AM é que, para GW, um dado vértice raiz $r \in V$ deve ser definido como parâmetro de entrada para a fase construtiva. Consequentemente, ao final da primeira fase, apenas a componente conexa que contenha r é submetida à fase de pruning. Como resultado, $|V|$ execuções do algoritmo de GW devem ser efetuadas para que o mesmo se torne um algoritmo de fator de aproximação 2 para a versão não roteada do problema. Contrariamente ao GW, a fase construtiva de AM, denominada em inglês de `UnrootedGrowthPhase`, não envolve um vértice raiz. Por este motivo, todas as componentes conexas resultantes da fase construtiva induzem árvores viáveis e podem ser submetidas à fase de pruning. Assim sendo, AM possui uma melhor complexidade computacional que GW; especificamente $O(n^2\log(n))$ contra $O(n^3\log(n))$. Outra diferença entre os dois algoritmos está na etapa de pós processamento. Por ser baseado em Programação Dinâmica, o procedimento de pruning introduzido em AM, denominado de `BestSubTree`, retorna sempre a melhor ASRP induzida pelas arestas da componente conexa sob investigação. Assim sendo, o pós processamento de AM domina o correspondente procedimento em GW.

Em função das vantagens descritas acima, o procedimento AM foi escolhido para ser empregado em nosso algoritmo NDRC. Essencialmente, a cada iteração k do MS, *custos complementares* $\{(1 - \overline{x}_e^k)c_e : e \in E\}$ e *penalidades complementares* $\{(1-\overline{x}_{i,n+1})d_i : i \in V\}$ são calculadas e utilizadas como parâmetros de entrada para o procedimento `UnrootedGrowthPhase`. Ao procedermos desta forma, tornamos as arestas presentes na solução $\overline{x}^k$ mais atraentes para escolha no algoritmo guloso. Como pode ser observado, empregamos informação dual Lagrangeana para guiar a construção de soluções viáveis para o problema. Espera-se com isto que ASRPs de boa qualidade sejam encontradas, na medida em que a solução dos PL(λ) se aproxima da solução ótima do PASRP.

Em nossa implementação, o algoritmo de Minkoff é chamado apenas nas iterações do MS nas quais observa-se uma melhora do melhor limite dual até então obtido para o problema. Em cada uma destas iterações, o procedimento `BestSubTree` é então chamado com custos e penalidades originais, ao invés dos corrrespondentes custos e penalidades complementares usados no procedimento `UnrootedGrowthPhase`. As

soluções obtidas ao final de `BestSubTree` são submetidas a um procedimento de Busca Local, descrito a seguir.

6.3.3.2 O procedimeto de Busca Local

Seja $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ uma ASRP. Em nosso procedimento de Busca Local, estamos essencialmente interessados em investigar árvores viáveis que diferem de $\mathcal{T}$ pela inclusão (resp., exclusão) de um vértice em (resp., de) $\mathcal{T}$. Contudo, não é difícil perceber que, para um dado vértice, digamos $i \in V$, a melhor maneira de inseri-lo em $\mathcal{T}$ pode envolver a remoção de vértices já presentes em $V_{\mathcal{T}}$. A mesma observação pode ser feita para a operação de exclusão de um vértice $j \in V_{\mathcal{T}}$. Assim sendo, empregamos em nosso procedimento de Busca Local uma vizinhança variável que envolve a investigação das ASRPs que resultam de uma operação *otimizada* de inclusão (resp. exclusão) de vértices em (resp. de) $\mathcal{T}$.

Em linhas gerais, esta versão otimizada de Busca Local funciona da seguinte forma. A inclusão de um vértice (resp. exclusão) é conduzida através de duas etapas. Inicialmente, uma árvore de mínimo custo conectando o conjunto expandido (resp. contraído) deve ser computada. Então, em uma segunda etapa, o procedimento `BestSubTree` deve ser aplicado à ASRP resultante. Somente após o término dessa segunda etapa, os benefícios do movimento de inclusão (resp. exclusão) podem ser avaliados.

Outro aspecto fundamental em nossa implementação consiste no esforço dispendido para manter baixos os tempos de CPU envolvidos na Busca Local. O objetivo primordial é que a BL possa ser usada a cada chamada do algoritmo AM. Para conseguir isto, diversos testes de dominância são realizados, de forma a evitar a avaliação completa de movimentos não *lucrativos*. Note que o tempo de computação exigido pelas duas etapas descritas é limitado (superiormente) pelo tempo necessário para se determinar uma AGM de G, isto é, $O(m\log(n))$. Para evitar pagar esses custos, implementamos alguns testes de dominância introduzidos no algoritmo de Busca Tabu para o PSG proposto por Ribeiro e Souza [113]. Como veremos a seguir, estes testes são facilmente adaptáveis para o PASRP.

6.3.3.3 Testes de dominância na Busca Local

Durante as iterações do nosso procedimento de Busca Local, assumimos que toda árvore $\mathcal{T}$ sob investigação satisfaz à uma certa propriedade, aqui denominada de

Condição de Otimalidade (CO) de $\mathcal{T}$ em relação ao grafo por ela induzido. Dizemos que $\mathcal{T}$ satisfaz a CO se não há uma ASRP com custo inferior ao custo de $\mathcal{T}$ no grafo induzido por $\mathcal{T}$. Assim sendo, se uma árvore que satisfaça a CO é utilizada como entrada para o algoritmo `BestSubTree`, a própria árvore deve ser obtida como resultado.

Inicialmente, vamos nos concentrar em testes de dominância relacionados aos movimentos de inserção de vértices. Considere então um vértice $i \notin V_{\mathcal{T}}$ e o conjunto de arestas que conectam i a $V_{\mathcal{T}}$, isto é, $\delta_{\mathcal{T}}(i) := \{e \in E : e \in \delta(i) \cap \delta(V_{\mathcal{T}})\}$. Assuma que as arestas em $\delta_{\mathcal{T}}(i)$, isto é, $\{e_1, e_2, ..., e_{|\delta_{\mathcal{T}}(i)|}\}$, sejam ordenadas em magnitude não decrescente de seus custos. Os seguintes testes podem ser aplicados para se avaliar os benefícios da inserção do vértice i em $\mathcal{T}$:

1. se $|\delta_{\mathcal{T}}(i)| = 0$, não há uma ASRP que conecte $V_{\mathcal{T}} \cup \{i\}$. O correspondente movimento é desconsiderado.

2. se $|\delta_{\mathcal{T}}(i)| = 1$, a inserção de i em T é lucrativa se e somente se $d_i > c_{e_1}$. Neste caso, já que a árvore inicial satisfaz a CO, a árvore resultante terá também essa propriedade.

3. se $|\delta(i)_{\mathcal{T}}| = 2$, devemos considerar dois casos:

   (a) Se $d_i > c_{e_1}$, a inserção de i é lucrativa e a árvore resultante de sua inserção satisfaz a CO.

   (b) Se $d_i \leq c_{e_1}$, a inserção de i ainda poderá ser lucrativa. Para entender as condições sob as quais isto pode ocorrer, vamos definir um *key-path* [34] entre dois vértices $k, j \in V_{\mathcal{T}}$ como uma aresta no caminho único entre k e j em $\mathcal{T}$, ou, alternativamente, como um caminho entre k e j no grafo induzido por $\mathcal{T}$, no qual todos os vértices internos (distintos das extremidades do caminho) possuem grau dois em $\mathcal{T}$. Note que, em ambos os casos, as extremidades do key-path podem ser distintas de j e k. O *peso líquido* do key-path corresponde à diferença entre a soma dos custos das arestas entre suas extremidades e a soma das penalidades de seus vértices internos. Observe que se introduzimos as arestas $e_1 = (i, k)$ e $e_2 = (j, k)$ e eliminarmos um key-path entre j e k, o grafo resultante induz uma ASRP. Para encontrar um key-path de peso máximo entre dois vértices de uma ARSP, implementamos um algoritmo baseado em

Programação Dinâmica, com custo proporcional a $O(|V_{\mathcal{T}}|)$. Suponha que um key-path de peso máximo tenha sido encontrado. Se $(c_{e_1} + c_{e_2} - d_i)$ for inferior ao peso deste key-path, a árvore que resulta da introdução de $\{e_1, e_2\}$ e da remoção das arestas e vértices internos deste key-path possui peso inferior ao peso da árvore inicial e satisfaz CO.

Caso as duas possibilidades de troca acima descritas forem lucrativas, aquela que produzir a maior redução de custos será efetivamente implementada.

4. se $|\delta_{\mathcal{T}}(i)| \geq 3$ procedemos da seguinte forma. Primeiro, inserimos a aresta e_1 em $\mathcal{T}$. Em seguida, tentamos reduzir o custo da árvore resultante, procurando introduzir arestas de $\{e_2, ..., e_{|\delta_{\mathcal{T}}(i)|}\}$ na ASRP récem obtida. Primeiro avaliamos (através de Custos Reduzidos de Programação Linear) a inserção de e_2, depois de e_3 e assim sucessivamente até que a inserção de $e_{|\delta_{\mathcal{T}}(i)|}$ seja avaliada. Sempre que a introdução de uma destas arestas for lucrativa, a inserção é realizada. Ao final deste processo, a ASRP resultante é submetida ao procedimento `BestSubTree`. Se a árvore retornada por este procedimento tiver peso inferior ao peso da árvore inicial (antes da inserção de e_1), a nova árvore substitui a anterior. Apesar da avaliação da inserção das arestas de $\{e_2, ..., e_{|\delta_{\mathcal{T}}(i)|}\}$ poder custar tempo proporcional a $O(|V_{\mathcal{T}}|^2)$, na prática, tempos computacionais bastante inferiores devem ser esperados. Isto se deve ao fato de que $|\delta_{\mathcal{T}}(i)|$ é normalmente muito inferior a $|V_{\mathcal{T}}|$ e ao uso eficiente das estruturas de dados propostas em [52] para avaliação de custos reduzidos de Programação Linear em árvores geradoras.

O procedimento de Busca Local é iniciado com a avaliação dos movimentos de inserção de vértices. Se esses movimentos não levam a uma árvore de menor peso, passamos à etapa seguinte da Busca Local, na qual movimentos de exclusão de vértices, mais caros, são avaliados.

Para entender os testes de dominância aplicados na avaliação de movimentos de exclusão de vértices de uma ASRP, vamos tratar por um instânte da eliminação de um vértice de Steiner j de uma árvore de Steiner $\mathcal{T}$ (veja [113] para detalhes). Assuma que $\{(i_1, j), (i_2, j), ..., (i_k, j)\}$ são as $k \geq 1$ arestas de $\mathcal{T}$ incidentes ao vértice $j \in V_{\mathcal{T}}$. A remoção de j e de suas arestas incidentes gera k sub-árvores de $\mathcal{T}$, identificadas por $\{\mathcal{T}_l : l = 1, \ldots, k\}$. Para cada subárvore $\mathcal{T}_l$, explicite o vértice $i_l \in T_l$ que divide uma aresta com j em $\mathcal{T}$. Assim sendo, os vértices correspondentes

$\{i_l : l = 1, \ldots, k\}$ são claramente identificados. Assuma que a aresta (p, q) seja a aresta de menor custo que conecta a sub-árvore $\mathcal{T}_l$ a qualquer outra das $k-1$ subárvores definidas acima. Foi provado (veja o Teorema 1 em [113]) que $\eta(j) := c_{(p,q)} - c_{(i_l,j)} - c_{(i_v,j)}$, onde $v \in \{1, \ldots, l-1, l+1, \ldots, k\}$ é um limite inferior para a variação de peso devida à eliminação do vértice j de $V_{\mathcal{T}}$. Naturalmente, sempre que $\eta(j) \geq 0$, a exclusão de um vértice de Steiner j não é lucrativa.

O resultado acima pode ser adaptado ao PASRP. Nesse caso, vamos redefinir $\eta(j)$ como $c_{(p,q)} - c_{(i_l,j)} - c_{(i_v,j)} + d_j$ e, assim sendo, é fácil perceber que $\eta(j)$ é um limite inferior para a variação de peso decorrente da eliminação de um vértice qualquer j de uma ASRP $\mathcal{T}$. Baseado nestes argumentos, antes de aplicar o procedimento de dois passos sugerido anteriormente (ou seja, a determinação de uma AGM seguido do procedimento `BestSubTree`) para avaliar o movimento de exclusão do vértice candidato j, determinamos o parâmetro $\eta(j)$ em tempo linear. Em nossa implementação, a escolha dos vértices i_l e i_v se dá tal qual sugerida em [113]. Por este critério, escolhemos i_v como o antecessor de j no caminho único entre a raiz de $\mathcal{T}$ e j. O vértice i_l, por sua vez, é tomado como a extremidade (distinta de i_v) da aresta de maior peso, indicente a j em $\mathcal{T}$.

6.4 Testes de redução e de fixação de variáveis

Antes de procedermos à obtenção de limites duais e primais como descrito na Seção anterior, aplicamos alguns testes que visam reduzir o tamanho de entrada dos problemas. Posteriormente, durante as iterações do MS, testes baseados em Custos Reduzidos de Programação Linear são também empregados para eliminar variáveis garantidamente sub-ótimas.

6.4.1 Testes de Redução

Os testes de redução aplicados ao PASRP procuram identificar vértices e arestas que, garantidamente, não estão presentes em nenhuma solução ótima para o problema. Até o momento, todos os testes disponíveis foram adaptados da literatura do PSG (veja Duin [33], por exemplo). Especificamente, dentre os testes que apresentaremos a seguir, os quatro primeiros foram adaptados em [92], o quinto em [85] e o último foi sugerido neste estudo. Esse conjunto de seis testes é aplicado repetidamente, em sequência, enquanto uma redução do grafo de entrada do problema for obtida por

pelo menos um deles.

6.4.1.1 Testes de caminho mínimo

O teste de caminho mínimo só é aplicado quando $c_{ij} > 0$, para toda aresta $(i, j) \in E$. Seja $\text{dist}(i, j)$ o custo do caminho mínimo entre os vértices $i, j \in V$. Se $\text{dist}(i, j) < c_{ij}$, então a aresta $(i, j) \in E$ pode ser eliminada de G.

6.4.1.2 Teste de grau unitário

Assuma que um dado vértice, digamos $i \in V$, possua grau um em G, ou seja, $|\delta(i)| = 1$, e denote por e a única aresta incidente a i naquele grafo. Se $c_e > d_i$, então o vértice i e, consequentemente, a aresta e não podem estar presentes em nenhuma ASRP ótima.

6.4.1.3 Teste de grau dois

Assuma que o vértice $i \in V$ possua grau dois em G, ou seja, $|\delta(i)| = 2$, e denote por (i, i_1) e (i, i_2) as duas arestas incidentes a i naquele grafo. Se $d_i = 0$ e as arestas (i, i_1) e (i, i_2) possuem custos positivos, duas possibilidades existem: ou ambas as arestas aparecem simultaneamente em uma ASRP ótima, ou nenhuma delas pode estar presente em uma solução ótima. Isso ocorre porque qualquer ASRP contendo i, com grau igual a um, leva, após a eliminação de i, a uma ASRP de menor peso, contradizendo assim uma possível hipótese de otimalidade.

Assim sendo, se $|\delta(i) = 2|$, o vértice i pode ser *pseudo-eliminado* por meio da introdução, em G, de uma aresta (i_1, i_2) com custo $c_{(i,i_1)} + c_{(i,i_2)}$. Se por ventura duas arestas incidentes simultaneamente a i_1, i_2 resultarem desta operação, apenas aquela de menor custo é preservada.

6.4.1.4 Teste de grau maior-que-dois

Quando todas as arestas indicentes a i possuem custos positivos, os testes anteriores podem ser estendidos a vértices $i \in V$ para os quais $d_i = 0$ e $|\delta(i)| \geq 3$. Por simplicidade, assuma que o vértice i, sob investigação satisfaça $|\delta(i)| = 3$. A idéia básica deste teste é que, sob certas condições, o vértice i só poderá aparecer com grau dois em uma ASRP ótima.

Para um vértice i como definido acima, considere as arestas $\{(i, i_1), (i, i_2), (i, i_3)\}$,

incidentes a i. Se,

$$\min\{\text{dist}(i_1,i_2)+\text{dist}(i_1,i_3), \text{dist}(i_2,i_1)+\text{dist}(i_2,i_3), \\ \text{dist}(i_3,i_1)+\text{dist}(i_3,i_2)\} \leq c_{e_1}+c_{e_2}+c_{e_3}, \quad (6.22)$$

não existe ASRP ótima onde i tenha grau 3 e o vértice poderá assim ser pseudo-eliminado de G. Como explicado anteriormente, isso é feito pela substituição de cada possível combinação de pares de arestas incidentes a i por uma aresta adequada, como no caso do teste de grau dois.

O teste acima pode ser generalizado para vértices $i \in V$ para os quais $|\delta(i)| > 3$ onde todas as arestas incidentes a i possuem custos positivos. Deve-se observar que o lado esquerdo de (6.22) fornece o custo de uma AGM do subgrafo de G induzido pelos vértices i_1, i_2 e i_3. Note, entretanto, que os pesos das arestas neste subgrafo são dados pelos custos de caminhos mínimos em G: $\text{dist}(i_1,i_2)$, $\text{dist}(i_1,i_3)$ e $\text{dist}(i_2,i_3)$. Repare que, quando $|\delta(i)| = 3$, esta AGM deve ter exatamente duas arestas. A generalização do teste inclui então o cálculo de AGMs (no subgrafo de G induzido por vértices adjacentes a i, onde os custos das arestas são os custos de caminhos mínimos entre suas extremidades em G) para toda possível combinação (com três ou mais elementos) de vértices incidentes a i. Analogamente a (6.22), para toda possível combinação de vértices (com 3 ou mais elementos), os custos dessas AGMs devem ser comparados ao custo da *estrela* que conecta i aos vértices a ele incidentes incluídos na combinação. Para que seja conclusivo, o custo de cada AGM não deve exceder o custo de cada estrela.

6.4.1.5 Testes de mínima adjacência

Considere uma aresta $(i,j) \in E$ conectando dois vértices $i, j \in V$ com penalidades positivas. Se $\min\{d_i, d_j\} - c_{(i,j)} > 0$ e $c_{(i,j)} = \min\{c_{(i,k)} : (i,k) \in E\}$, os vértices i e j podem ser fundidos em um único vértice com penalidade $d_i + d_j - c_{(i,j)}$. Neste caso, sempre que duas arestas (i,l) e (j,l) existirem, duas *arestas paralelas* conectando l ao vértice $\{i,j\}$ resultariam da fusão. Nesse caso, apenas a aresta de menor custo deve ser mantida.

6.4.1.6 Teste de ganho líquido de comprimento dois (TGLC2)

Considere três vértices em V, digamos i, j, e k e as arestas (i,j), (i,k), e (j,k). Assuma que as condições $c_{(i,k)} + c_{(j,k)} - d_k < c_{(i,j)}$ e $c_{(i,j)} \geq \min\{c_{(i,k)}; c_{(j,k)}\}$ sejam

satisfeitas. Então, a aresta (i, j) é necessariamente subótima. Isto ocorre porque a aresta (i, j) não é necessária para definir uma AGM envolvendo $\{i, j, k\}$ e porque ao invés de conectarmos os vértices i e j por meio da aresta (i, j), podemos fazê-lo de forma mais econômica através do caminho dado pelas arestas (i, k) e (j, k), que se beneficia da penalidade d_k.

Na Figura 6.3, ilustramos um caso onde o TGLC2 permite a eliminação da aresta (i, j). Verifique que, para este exemplo, $\text{dist}(i, j) = c_{(i,j)}$. Assim sendo, o TGLC2 aprimora o teste de distância mínima.

Já na Figura 6.4, indicamos um caso para o qual o TGLC2 não permite a eliminação da aresta (i, j). Note que, apesar do caminho alternativo entre i e j ser lucrativo (isto é, $c_{(i,k)} + c_{(j,k)} - d_k < c_{(i,j)}$), a condição $c_{(i,j)} \geq \min\{c_{(i,k)}; c_{(j,k)}\}$, por não ser satisfeita, implica na necessidade da aresta (i, j) para construção de uma árvore geradora mínima envolvendo i, j, k.

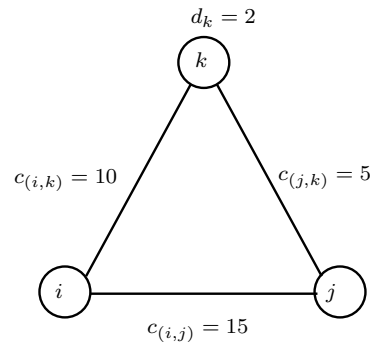

Figura 6.3: Exemplo onde o TGLC2 elimina a aresta (i, j)

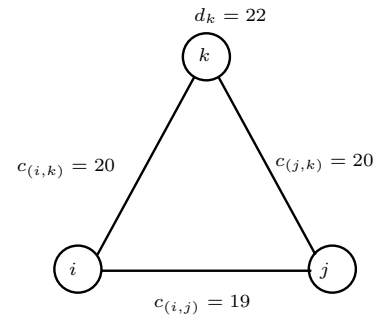

Figura 6.4: Exemplo onde o TGLC2 não elimina a aresta (i, j)

6.4.2 Testes de fixação de variáveis

Além dos testes de redução que acabamos de descrever, utilizamos também testes de fixação de variáveis. Estes, por sua vez, são aplicados durante as iterações do

MS, através da determinação de CRPL (vide Seção 4.2.3, Capítulo 4).

Cada um dos Problemas Lagrangeanos LP(λ) definidos na Seção 6.3.1 corresponde ao problema de determinar, ao mínimo custo Lagrangeano, uma floresta com exatas $|V'|-2$ arestas de G'. Portanto, a determinação dos CRPL correspondentes é bastante simples. Estes podem ser calculados através de trocas convenientes de arestas, que resultam diretamente das trocas de arestas necessárias para computar CRPL de AGMs, conforme descrito em detalhes na Seção 4.2.3 do Capítulo 4.

Para entender como estes testes são aplicados ao problema aqui tratado, assuma que $\overline{x}^k$ seja a solução ótima de PL(λ^k). Para esta solução, assuma também que $\overline{c}_e^k$, onde $e=(i,j)\in E'$, seja o CRPL associado à variável x_e. Então, $\overline{c}_e^k$ pode ser computado como

$$\overline{c}_e^k = c_e^k - c_{e_0}^k, \tag{6.23}$$

onde e_0 é uma aresta que depende de i e j estarem ou não na mesma componente conexa da floresta induzida por $\overline{x}^k$. Sempre que i e j estiverem na mesma componente conexa, e_0 é a aresta de máximo custo Lagrangeano no caminho único que conecta i e j através de arestas naquela componente. Alternativamente, se i e j encontram-se em componentes conexas distintas, e_0 é a aresta de maior custo Lagrangeano da floresta.

Uma vez computado $\overline{c}_e^k$, se $\overline{c}_e^k + w_{\lambda^k} > \overline{w}$ (onde $\overline{w}$ denota um limite superior válido para o PASRP), a variável x_e tem a garantia de não pertencer a nenhuma ASRP ótima. Neste caso, a variável poderá ser *fixada em zero* e a aresta e poderá ser eliminada de G'. Salientamos também que, diante de custos e penalidades inteiros, a condição $\lceil \overline{c}_e^k + w_{\lambda^k} \rceil > \overline{w}$ é suficiente para eliminar variáveis de qualquer ASRP ótima.

6.5 Experimentos computacionais com a Heurística Lagrangena

Nesta Seção, descrevemos os experimentos computacionais conduzidos para avaliar a Heurística Lagrangeana introduzida neste Capítulo. Inicialmente, descrevemos os conjuntos de instâncias empregados nos experimentos. Posteriormente, apresentamos e comentamos os resultados.

6.5.1 Conjuntos de instâncias testados

Seis conjuntos de instâncias foram testados em nossos experimentos computacionais. No total, 168 instâncias foram empregadas para avaliar os métodos propostos.

Os dois primeiros conjuntos de instâncias que utilizamos foram introduzidos em [70]. O primeiro deles, P, contém 11 instâncias. Estas, por sua vez, apresentam uma relação aproximadamente constante de grau e penalidades por peso. Já as 23 instâncias do segundo conjunto, K, induzem grafos com topologia similar a mapas de cidades.

O terceiro e quarto conjuntos de instâncias, C e D, foram gerados em [18] a partir das instâncias C e D do PSG introduzidas em [9]. Cada instância do PSG deu origem a duas instâncias do PASRP. Isto foi feito através da atribuição de penalidades inteiras sorteadas uniformemente no intervalo $[1, max]$ para cada vértice terminal em cada instância do PSG. Para tanto, dois valores distintos de max foram empregados, um dando origem à série A de instâncias e o outro levando à série B. Para as instâncias da série A, o valor de max foi fixado em 10, enquanto que para as instâncias da série B, aquele valor foi fixado em 100. Os vértices não terminais das instâncias do PSG, por sua vez, deram origem a vértices com penalidades nulas nas respectivas instâncias do PASRP. Os custos das arestas nas instâncias do PASRP são iguais aos custos das arestas que lhes deram origem no PSG. As 40 instâncias do quinto conjunto, E, introduzidas em [85], foram geradas de forma semelhante, a partir das instâncias E do PSG, também introduzidas em [9].

O último conjunto de instâncias, H, foi gerado em [85] a partir das instâncias hipercubo propostas para o PSG em [115]. Penalidades inteiras, aleatoriamente escolhidas nos intervalos $[1, 2]$ e $[100, 200]$, foram atribuídas aos vértices terminais das instâncias de origem, gerando duas séries de instâncias para o PASRP, respectivamente as séries u e p. Vértices com penalidades nulas foram gerados a partir dos vértices não terminais das instâncias do PSG. Os grafos de definição destes problemas são hipercubos d-dimensionais, onde $d \in \{6, ..., 12\}$ e 2^d vértices e $d2^{d-1}$ arestas existem. Os grafos de definição destas instâncias são bipartidos e as partições correspondem ao conjunto de vértices com penalidades positivas e nulas.

6.5.2 Resultados computacionais da heurística Lagrangeana

Os algoritmos propostos neste Capítulo foram implementados em linguagem C. Os resultados apresentados nesta Seção foram obtidos através de uma máquina com processador Intel Pentium IV de 3GHz e 512 Mbytes de memória RAM. O compilador GNU `gcc` foi empregado com chave de otimização `-O3` ativada, no sistema operacional Linux.

Na Tabela 6.1, apresentamos uma síntese dos resultados obtidos com os testes de pré-processamento. Esta Tabela resume os resultados detalhados apresentados nas Tabelas 6.2 - 6.5 para cada conjunto de instâncias. Nas Tabelas referentes a resultados de pré-processamento, denotamos por $|V'|$ e $|E'|$ o número de vértices e arestas de cada instância após a aplicação dos testes. A primeira coluna da Tabela 6.1 indica o conjunto de instâncias considerado. Nas colunas seguintes, apresentamos dois grupos de resultados referentes a duas simulações distintas de pré-processamento. Para o primeiro grupo (colunas 2-4), a simulação incluiu todos os testes da Seção 6.4.1. Para a segunda simulação (colunas 5-7), o TGLC2 não foi aplicado. Para cada simulação, apresentamos a taxa média de redução do número de vértices e arestas, indicado por $\frac{|V'|}{|V|}$ e $\frac{|E'|}{|E|}$, respectivamente e o tempo médio de CPU (em segundos) necessário para a execução dos testes. Informações similares, detalhadas para cada instância de cada conjunto, são apresentadas nas Tabelas 6.2 - 6.5.

Tabela 6.1: Consolidação de resultados de pré-processamento

	Todos os testes de pré-processamento			Todos os testes, exceto o TGLC2		
	$\frac{\lvert V'\rvert}{\lvert V\rvert}$	$\frac{\lvert E'\rvert}{\lvert E\rvert}$	t(s)	$\frac{\lvert V'\rvert}{\lvert V\rvert}$	$\frac{\lvert E'\rvert}{\lvert E\rvert}$	t(s)
P	0.84	0.79	0.17	0.84	0.79	0.14
K	0.44	0.38	0.23	0.47	0.50	0.27
C	0.70	0.39	0.75	0.71	0.42	1.17
D	0.72	0.44	3.04	0.72	0.46	4.31
E	0.72	0.20	33.0	0.72	0.21	31.98

Tabela 6.2: Resultados do pré-processamento - Instâncias P e K

			Após					
			Todos os testes de pré-processamento			Todos os testes, exceto o TGLC2		
Instância	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$	t(s)	$\|V'\|$	$\|E'\|$	t(s)
P100	100	317	66	159	0.01	66	163	0.01
P100.1		284	87	200	0.01	87	200	0.01
P100.2		297	78	192	0.01	78	192	0.01
P100.3		316	91	235	0.01	91	236	0.01
P100.4		284	69	176	0.01	69	181	0.02
P200	200	587	167	438	0.05	167	440	0.05
P400	400	1200	347	1007	0.23	347	1008	0.23
P400.1		1212	325	980	0.30	325	986	0.30
P400.2		1196	341	994	0.41	341	997	0.32
P400.3		1175	342	980	0.41	342	983	0.31
P400.4		1144	344	943	0.41	344	946	0.32
K100	100	351	36	109	0.03	40	169	0.03
K100.1		348	35	112	0.05	35	120	0.05
K100.2		339	27	86	0.04	27	96	0.05
K100.3		407	18	65	0.25	19	86	0.24
K100.4		364	30	81	0.02	36	132	0.02
K100.5		358	36	118	0.03	36	135	0.03
K100.6		307	28	76	0.01	28	80	0.01
K100.7		315	21	51	0.01	23	65	0.01
K100.8		343	42	130	0.02	43	144	0.02
K100.9		333	17	37	0.01	20	64	0.01
K100.10		319	36	93	0.02	37	111	0.02
K200	200	691	67	193	0.23	85	295	0.39
K400	400	1515	202	663	0.40	222	865	0.39
K400.1		1470	184	562	0.31	201	762	0.26
K400.2		1527	186	592	3.28	212	897	4.22
K400.3		1492	173	537	0.32	184	643	0.31
K400.4		1426	159	490	0.26	181	717	0.22
K400.5		1456	179	495	0.32	204	701	0.33
K400.6		1576	215	745	0.42	236	972	0.46
K400.7		1442	208	609	0.29	215	810	0.32
K400.8		1516	227	849	0.28	236	989	0.29
K400.9		1500	181	601	0.54	189	746	0.76
K400.10		1507	181	621	0.42	196	795	0.55

Tabela 6.3: Resultados do pré-processamento - Instâncias C

			Após					
			Todos os testes de pré-processamento			Todos os testes, exceto o TGLC2		
Instância	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$	t(s)	$\|V'\|$	$\|E'\|$	t(s)
c1A	500	625	116	214	0.17	116	214	0.17
c1B			125	226	0.17	125	226	0.17
c2A			110	208	0.17	110	208	0.16
c2B			111	209	0.17	111	209	0.18
c3A			167	284	0.21	167	285	0.20
c3B			185	303	0.23	185	304	0.24
c4A			198	321	0.24	198	321	0.24
c4B			222	345	0.30	222	345	0.29
c5A			210	326	0.26	210	328	0.26
c5B			205	317	0.32	205	322	0.29
c6A	500	1000	356	823	0.18	356	823	0.18
c6B			356	823	0.18	356	823	0.18
c7A			366	843	0.18	366	843	0.18
c7B			365	842	0.27	365	842	0.28
c8A			372	853	0.28	372	855	0.28
c8B			369	848	0.38	369	850	0.38
c9A			400	889	0.42	400	891	0.42
c9B			392	881	0.40	392	882	0.41
c10A			385	865	0.51	385	871	0.41
c10B			327	789	0.50	327	804	0.43
c11A	500	2500	489	2143	0.21	489	2143	0.22
c11B			489	2143	0.22	489	2143	0.21
c12A			484	2185	0.40	484	2186	0.40
c12B			484	2185	0.41	484	2186	0.41
c13A			473	2104	0.76	473	2120	0.58
c13B			472	2095	0.76	472	2119	0.58
c14A			466	2052	0.75	466	2081	0.57
c14B			458	2006	0.74	458	2044	0.57
c15A			406	1808	0.77	405	1861	0.65
c15B			369	1632	0.70	368	1735	0.59
c16A	500	12500	500	4730	0.92	500	4740	0.49
c16B			500	4729	0.93	500	4740	0.49
c17A			498	4639	2.79	498	4685	1.66
c17B			498	4638	2.84	498	4685	1.66
c18A			468	3860	2.28	468	4318	2.04
c18B			464	3761	2.35	464	4270	2.11
c19A			426	2790	1.75	433	3706	1.61
c19B			389	2059	1.93	412	3150	2.26
c20A			287	1415	1.63	316	2631	9.07
c20B			7	10	1.41	185	1172	15.47

Tabela 6.4: Resultados do pré-processamento - Instâncias D

			Após					
			Todos os testes de pré-processamento			Todos os testes, exceto o TGLC2		
Instância	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$	t(s)	$\|V'\|$	$\|E'\|$	t(s)
d1A	1000	1250	233	443	0.74	233	443	0.72
d1B			233	443	0.74	233	443	0.73
d2A			261	485	0.73	261	485	0.73
d2B			264	488	0.74	264	488	0.73
d3A			333	564	0.82	333	564	0.81
d3B			378	612	0.94	378	612	0.94
d4A			358	591	0.93	358	591	0.91
d4B			395	629	0.97	395	629	0.97
d5A			444	687	1.16	444	688	1.16
d5B			418	654	1.31	418	657	1.30
d6A	1000	2000	741	1708	0.79	741	1708	0.77
d6B			741	1708	0.78	741	1708	0.78
d7A			735	1706	0.79	735	1706	0.77
d7B			736	1707	0.78	736	1707	0.77
d8A			786	1764	1.74	786	1764	1.72
d8B			780	1758	1.72	780	1759	1.72
d9A			775	1738	1.72	775	1741	1.24
d9B			761	1718	1.68	761	1722	1.68
d10A			733	1698	2.47	733	1704	2.05
d10B			635	1580	2.43	635	1593	2.11
d11A	1000	5000	986	4658	0.90	986	4658	0.89
d11B			986	4658	0.89	986	4658	0.88
d12A			991	4639	1.75	991	4639	1.75
d12B			991	4639	1.75	991	4639	1.74
d13A			965	4517	4.12	965	4541	3.33
d13B			960	4503	4.10	960	4532	3.30
d14A			946	4470	4.04	946	4499	3.27
d14B			930	4429	3.93	930	4467	3.19
d15A			841	4076	4.11	841	4200	3.45
d15B			748	3696	3.54	748	3891	3.04
d16A	1000	25000	1000	10588	4.36	1000	10595	2.75
d16B			1000	10588	4.31	1000	10595	2.70
d17A			999	10518	8.78	999	10531	5.36
d17B			999	10518	7.88	999	10531	5.30
d18A			946	9089	7.46	946	9601	6.31
d18B			928	8586	6.80	928	9256	6.52
d19A			897	7958	8.39	897	8906	8.06
d19B			861	7020	8.42	861	8248	8.60
d20A			652	4096	6.44	658	5872	10.96
d20B			276	879	5.66	413	2752	68.23

Tabela 6.5: Resultados do pré-processamento - Instâncias E

			Após					
			Todos os testes de pré-processamento			Todos os testes, exceto o TGLC2		
Instância	$\|V\|$	$\|E\|$	$\|V'\|$	$\|E'\|$	t(s)	$\|V'\|$	$\|E'\|$	t(s)
e1A	2500	3125	653	1248	5.21	653	1248	5.26
e1B			655	1250	5.18	655	1250	5.48
e2A			695	1305	5.15	695	1305	5.38
e2B			697	1307	5.13	697	1307	5.64
e3A			885	1495	6.41	885	1495	6.73
e3B			970	1582	7.51	970	1582	7.66
e4A			951	1558	6.66	951	1558	6.72
e4B			991	1599	7.53	991	1599	7.52
e5A			1095	1712	9.02	1096	1714	9.00
e5B			1053	1668	10.05	1053	1669	9.93
e6A	2500	5000	1821	4283	5.62	1821	4283	5.60
e6B			1821	4283	5.61	1821	4283	6.02
e7A			1864	4340	5.63	1864	4340	5.69
e7B			1865	4341	5.65	1865	4341	5.63
e8A			1928	4406	11.94	1928	4406	12.17
e8B			1914	4390	12.29	1914	4390	12.09
e9A			1955	4431	12.71	1955	4435	12.32
e9B			1924	4399	12.52	1924	4404	11.96
e10A			1816	4279	17.83	1816	4289	17.12
e10B			1610	4042	18.53	1610	4060	15.00
e11A	2500	12500	2491	12063	8.30	2491	12063	7.78
e11B			2491	12063	9.85	2491	12063	8.35
e12A			2490	12089	15.62	2490	12090	8.55
e12B			2490	12089	15.37	2490	12090	7.87
e13A			2430	11897	28.76	2430	11923	22.29
e13B			2407	11847	29.72	2407	11880	22.70
e14A			2369	11844	35.22	2369	11876	27.88
e14B			2311	11695	34.08	2311	11747	27.86
e15A			2055	10755	37.74	2056	10919	29.38
e15B			1862	9993	31.97	1862	10279	26.04
e16A	2500	62500	2500	29331	66.42	2500	29332	43.54
e16B			2500	29331	62.88	2500	29332	48.42
e17A			2500	29086	61.30	2500	29090	44.66
e17B			2500	29084	63.36	2500	29090	47.52
e18A			2378	27044	113.56	2378	27680	105.24
e18B			2347	26508	113.98	2347	27304	99.09
e19A			2180	20845	132.68	2183	23005	128.51
e19B			2031	13168	127.46	2075	18283	203.10
e20A			1652	12884	83.36	1661	17044	89.76
e20B			779	2825	72.53	1171	8398	108.09

Os resultados apresentados na Tabela 6.1 indicam que o uso do TGLC2 foi bastante positivo para as instâncias do conjunto K. Note que para este conjunto, a taxa média de eliminação de arestas cresceu em 24% em função do novo teste. Dentre as instâncias dos conjuntos C, D e E, as que mais se beneficiaram do TGLC2 foram, confirmando expectativas, aquelas instâncias que possuem maior proporção de vértices com penalidades positivas. Para as instâncias em que a taxa de eliminação de arestas foi indiferente à introdução de TGLC2, não houve, de um modo geral, crescimento apreciável dos respectivos tempos de CPU, com a introdução do referido teste. Na realidade, para os conjuntos de instâncias C e E, até mesmo uma redução do tempo total médio de CPU foi observado, com o emprego do teste. É bem verdade que este resultado foi influenciado fortemente pelas reduções de tempos de CPU para as maiores instâncias destes conjuntos: C20A, C20B, D20A, D20B. Por fim, os testes aplicados neste estudo se mostraram incapazes de reduzir as dimensões das instâncias do conjunto H.

Nas Tabelas 6.7 a 6.11, apresentamos os resultados da heurística Lagrangeana para cada conjunto de instâncias consideradas em nosso estudo. Os resultados apresentados nestas Tabelas foram obtidos ao se empregar os seguintes parâmetros de controle no Método do Subgradiente: SMMAX = 2000 iterações, frequência de redução $\xi = 100$ e IDADEMAX = 1 (isto é, a sobrevida das GSECs não foi estendida).

A primeira coluna em cada uma das Tabelas 6.7 a 6.11 indica a instância do conjunto. Nas Tabelas 6.7 a 6.10, as duas colunas seguintes indicam os melhores limites inferiores e superiores obtidos pela heurística, ou seja, w_d e $\overline{w}$, respectivamente. Na coluna seguinte, é indicado o correspondente gap de dualidade $\frac{\overline{w}-w_d}{w_d}$, em valores percentuais. Nas duas colunas finais, apresentamos o tempo de CPU (em segundos) e w, o valor da função objetivo ótima.

Na Tabela 6.11, referente ao conjunto H, apresentamos algumas informações adicionais às apresentadas nas Tabelas 6.7 - 6.10. São elas: as dimensões das instâncias, na segunda e terceria colunas e o tempo de CPU referente aos testes de pré-processamento (TGLC2 incluso), na oitava coluna. Na última coluna desta Tabela, apresentamos também o melhor limite superior conhecido para cada instância. Uma indicação "*"nesta coluna indica que o melhor limite primal iguala w. Já uma indicação "+"indica que o correspondente melhor limite primal foi obtido pela nossa heurística Lagrangeana.

Os resultados das Tabelas 6.7 a 6.11 são sintetizados na Tabela 6.6, logo a seguir.

As quatro primeiras colunas desta Tabela indicam, respectivamente, o conjunto de instâncias, o número total de instâncias em cada conjunto, o número de instâncias no conjunto para as quais nossa heurística provou a otimalidade do limite superior obtido ($w_d = \overline{w}$) e o número de instâncias no conjunto para as quais a nossa heurística encontrou a solução ótima ($\overline{w} = w$). As duas colunas finais indicam, respectivamente, os gaps de dualidade e os tempos médios de CPU (em segundos) para cada conjunto de instâncias.

Tabela 6.6: Resultados consolidados da Heurística Lagrangeana

	Número de Instâncias				
	no grupo	resolvidas	$\overline{w} = w$	gap [%]	t(s)
P	11	5	11	0.324	3.053
K	23	11	22	3.829	1.540
C	40	17	40	0.916	5.914
D	40	9	40	1.256	34.832
E	40	6	36	1.480	427.775
H ($d \leq 10$)	10	-	-	6.746	21.630
H ($d = 11, 12$)	4	-	-	10.46	1603.480

Como pode ser observado, para as instâncias nos conjuntos P, K, C, D e E nossos limites primais são sempre muito bons. Das 154 instâncias nestes conjuntos, nossa heurística encontrou a solução ótima para 149 delas. Para as 5 restantes, a diferença máxima entre o limite primal obtido e o correspondente valor ótimo foi de 1.49%.

Para as instâncias dos conjuntos P, K, C e D (as instâncias do conjunto E não foram testadas em [18]), os resultados obtidos por nossa heurística Lagrangeana se comparam favoravelmente com aqueles obtidos pela Metaheurística em [18]. Para as 114 instâncias nestes conjuntos, o nosso algoritmo encontrou a solução ótima em 113 casos, enquanto que os algoritmos em [18] obtiveram êxito em 91 casos. Embora os algoritmos propostos em [18] não empreguem os testes de pré-processamento aqui utilizados (e, portanto, provavelmente possam se tornar ainda mais eficientes), acreditamos que nosso algoritmo seja também competitivo com aqueles propostos em [18] em termos de tempos de CPU. Naquela referência, os tempos médios de processamento, obtidos em uma máquina IBM Pentium II com 400 Mhz e 64 Mbytes de memória RAM, para as instâncias P,K, C e D foram de 202.45, 71.52, 875.77 e 5847.93 segundos, respectivamente. Comparando estes valores com os tempos médios de CPU empregados por nossa heurística Lagrangena para os mesmos conjuntos

Tabela 6.7: Resultados da Heurística Lagrangeana - Instâncias P e K

Instância	w_d	$\overline{w}$	gap [%]	t(s)	w
P100	803300.0000	803300	Opt	0.25	803300
P100.1	926238.0000	926238	Opt	0.32	926238
P100.2	401641.0000	401641	Opt	0.31	401641
P100.3	659644.0000	659644	Opt	0.23	659644
P100.4	827419.0000	827419	Opt	0.18	827419
P200	1311970.4538	1317874	0.450	1.58	1317874
P400	2450885.8128	2459904	0.368	5.74	2459904
P400.1	2776739.1010	2808440	1.142	7.52	2808440
P400.2	2512006.9007	2518577	0.262	5.18	2518577
P400.3	2938688.2638	2951725	0.444	5.99	2951725
P400.4	2827694.8332	2852956	0.893	6.28	2852956
K100	135511.0000	135511	Opt	0.11	135511
K100.1	124108.0000	124108	Opt	0.09	124108
K100.2	200262.0000	200262	Opt	0.18	200262
K100.3	115953.0000	115953	Opt	0.25	115953
K100.4	87498.0000	87498	Opt	0.04	87498
K100.5	119078.0000	119078	Opt	0.06	119078
K100.6	132886.0000	132886	Opt	0.02	132886
K100.7	172457.0000	172457	Opt	0.10	172457
K100.8	210869.0000	210869	Opt	0.34	172457
K100.9	122917.0000	122917	Opt	0.02	122917
K100.10	133567.0000	133567	Opt	0.03	133567
K200	316673.9170	329211	3.959	0.74	329211
K400	326913.4753	350093	7.090	2.73	350093
K400.1	448931.0463	490771	9.320	2.75	490771
K400.2	426979.2349	477073	11.732	5.57	477093
K400.3	391085.1170	415328	6.199	2.39	415328
K400.4	366707.8591	389451	6.202	2.07	389451
K400.5	495385.8162	519526	4.873	2.33	519526
K400.6	352497.1554	374849	6.341	3.34	374849
K400.7	441518.7576	474466	7.462	3.07	474466
K400.8	398891.5238	418614	4.944	3.42	418614
K400.9	357336.0506	383105	7.211	3.00	383105
K400.10	350741.3267	395413	12.736	2.76	394191

Tabela 6.8: Resultados da Heurística Lagrangeana - Instâncias C

Instância	w_d	$\overline{w}$	gap [%]	t(s)	w
c1A	18.0000	18	Opt	0.18	18
c1B	83.0874	85	2.302	0.84	85
c2A	49.4865	50	Opt	0.18	50
c2B	139.4014	141	1.147	0.85	141
c3A	413.1276	414	Opt	0.50	414
c3B	733.5273	737	0.473	2.16	737
c4A	616.9955	618	0.163	1.60	618
c4B	1054.3364	1063	0.822	3.49	1063
c5A	1079.0071	1080	Opt	1.48	1080
c5B	1525.6234	1528	0.156	2.56	1528
c6A	17.5147	18	Opt	0.21	18
c6B	50.4103	55	9.105	3.28	55
c7A	49.2453	50	Opt	0.38	50
c7B	101.0692	102	Opt	2.36	102
c8A	359.7694	361	0.342	4.78	361
c8B	496.5290	500	0.699	6.52	500
c9A	530.0124	533	0.564	6.57	533
c9B	689.9805	694	0.583	10.28	694
c10A	856.9452	859	0.240	5.90	859
c10B	1067.1803	1069	0.171	6.67	1069
c11A	17.0926	18	Opt	1.00	18
c11B	29.7161	32	7.686	5.50	32
c12A	37.0331	38	Opt	2.24	38
c12B	43.8827	46	4.825	6.38	46
c13A	234.7131	236	0.548	11.08	236
c13B	255.9738	258	0.792	14.71	258
c14A	290.1036	293	0.998	13.20	293
c14B	315.1460	318	0.906	13.69	318
c15A	498.1160	501	0.579	16.49	501
c15B	549.0151	551	0.362	14.89	551
c16A	10.0490	11	Opt	5.09	11
c16B	10.5539	11	Opt	9.38	11
c17A	17.0040	18	Opt	7.84	18
c17B	17.0590	18	Opt	7.80	18
c18A	109.3350	111	1.523	15.52	111
c18B	111.1436	113	1.670	16.75	113
c19A	145.0291	146	Opt	5.58	146
c19B	145.0766	146	Opt	4.33	146
c20A	265.0154	266	Opt	2.88	266
c20B	266.0868	267	Opt	1.41	267

Tabela 6.9: Resultados da Heurística Lagrangeana - Instâncias D

Instância	w_d	$\overline{w}$	gap [%]	t(s)	w
d1A	18.0000	18	Opt	0.75	18
d1B	104.0882	106	1.837	2.34	106
d2A	49.2103	50	Opt	0.74	50
d2B	217.0053	218	Opt	1.98	218
d3A	806.0212	807	Opt	2.75	807
d3B	1504.1409	1509	0.323	8.21	1509
d4A	1200.0942	1203	0.242	5.10	1203
d4B	1878.0088	1881	0.159	7.46	1881
d5A	2155.0896	2157	0.089	8.07	2157
d5B	3124.1323	3135	0.348	10.16	3135
d6A	17.2068	18	Opt	0.93	18
d6B	61.6590	67	8.662	11.40	67
d7A	49.2524	50	Opt	1.78	50
d7B	100.7147	103	2.269	11.11	103
d8A	751.9670	755	0.403	17.84	755
d8B	1031.7074	1036	0.416	33.46	1036
d9A	1065.6080	1070	0.412	22.52	1070
d9B	1415.4595	1420	0.321	47.73	1420
d10A	1668.1245	1671	0.172	37.11	1671
d10B	2074.9530	2079	0.195	43.25	2079
d11A	17.2934	18	Opt	4.82	18
d11B	25.9028	29	11.957	20.58	29
d12A	40.4884	42	3.733	28.64	42
d12B	40.6799	42	3.245	23.98	42
d13A	442.7795	445	0.501	58.28	445
d13B	483.9459	486	0.424	81.47	486
d14A	598.2902	602	0.620	86.95	602
d14B	662.3140	665	0.406	121.85	665
d15A	1038.9896	1042	0.290	105.53	1042
d15B	1105.9666	1108	0.184	92.06	1108
d16A	12.0134	13	Opt	22.83	13
d16B	12.0453	13	Opt	23.47	13
d17A	21.8721	23	5.157	43.90	23
d17B	21.8639	23	5.196	47.58	23
d18A	216.1264	218	0.867	76.55	218
d18B	221.5386	223	0.660	71.48	223
d19A	304.9376	306	0.348	89.42	306
d19B	308.6635	310	0.433	71.23	310
d20A	534.9921	536	0.188	38.75	536
d20B	535.9886	537	0.189	9.21	537

Tabela 6.10: Resultados da Heurística Lagrangeana - Instâncias E

Instância	w_d	$\overline{w}$	gap [%]	t(s)	w
e01A	13.0000	13	Opt	5.25	13
e01B	105.1852	109	3.627	13.81	109
e02A	30.0000	30	Opt	5.25	30
e02B	164.3261	170	3.453	16.25	170
e03A	2227.8590	2231	0.141	24.81	2231
e03B	3796.3260	3806	0.255	82.52	3806
e04A	3146.5045	3151	0.143	35.57	3151
e04B	4876.7883	4888	0.230	94.24	4888
e05A	5651.8605	5657	0.091	57.24	5657
e05B	7990.8468	7998	0.090	103.73	7998
e06A	18.3809	19	Opt	5.79	19
e06B	67.3600	70	3.919	66.35	70
e07A	39.3128	40	Opt	8.34	40
e07B	130.2630	136	4.404	70.47	136
e08A	1872.8424	1878	0.275	152.55	1878
e08B	2547.5657	2555	0.292	424.04	2555
e09A	2781.8277	2787	0.186	196.88	2787
e09B	3535.8632	3541	0.145	761.16	3541
e10A	4580.8285	4586	0.113	439.38	4586
e10B	5496.8508	5502	0.094	696.69	5502
e11A	20.2059	21	Opt	67.74	21
e11B	31.6056	34	7.576	190.08	34
e12A	48.0288	49	Opt	164.91	49
e12B	64.1235	68	6.045	214.32	67
e13A	1167.0608	1169	0.166	933.14	1169
e13B	1266.1192	1270	0.307	1075.74	1269
e14A	1575.6258	1579	0.214	875.29	1579
e14B	1712.9595	1716	0.177	1732.24	1716
e15A	2607.9719	2610	0.078	1355.50	2610
e15B	2764.9737	2767	0.073	1315.38	2767
e16A	13.9277	15	7.699	330.50	15
e16B	13.9762	15	7.325	341.25	15
e17A	23.7220	25	5.387	371.07	25
e17B	23.8987	25	4.608	373.75	25
e18A	552.5118	557	0.812	1264.09	555
e18B	562.0238	566	0.707	1274.03	564
e19A	745.6350	747	0.183	908.66	747
e19B	756.1145	758	0.249	528.91	758
e20A	1329.9984	1331	0.075	429.85	1331
e20B	1340.9730	1342	0.077	104.21	1342

Tabela 6.11: Resultados da Heurística Lagrangeana - Instâncias H

Instância	$\|V\|$	$\|E\|$	w_d	$\overline{w}$	gap [%]	t(s) total	t(s) pre-proc	melhor $\overline{w}$ conhecido
hc6p	64	192	3822.49	3985	4.25	0.50	0.00	3908 (*)
hc6u	64	192	35.46	37	4.33	0.48	0.00	36 (*)
hc7p	128	448	7551.41	8134	7.72	1.49	0.01	7739
hc7u	128	448	70.48	73	3.57	1.17	0.00	72 (*)
hc8p	256	1024	14955.75	16023	7.13	5.20	0.05	15274
hc8u	256	1024	140.43	151	7.53	4.14	0.04	150
hc9p	512	2304	29598.42	32151	8.62	16.06	0.22	32151 (+)
hc9u	512	2304	278.78	296	6.18	13.13	0.19	296 (+)
hc10p	1024	5120	58865.31	64974	10.38	114.36	0.95	64974 (+)
hc10u	1024	5120	551.28	594	7.75	59.77	0.81	594
hc11p	2048	11264	116751.68	130252	11.56	629.92	5.24	130252 (+)
hc11u	2048	11264	1099.82	1199	9.02	360.61	4.41	1199 (+)
hc12p	4096	24576	231727.31	257833	11.27	3507.70	30.18	257833 (+)
hc12u	4096	24576	2184.48	2403	10.00	1915.72	27.56	2403 (+)

de instâncias (veja a última coluna da Tabela 6.6), verificamos que os algoritmos em [18] gastaram entre 46 e 167 vezes mais tempo de CPU que nossa heurística. Acreditamos que estas razões de tempo são compatíveis com as diferenças de computadores empregados nos respectivos experimentos computacionais. Assim sendo, estes resultados sugerem que nosso algoritmo não é dominado em tempo de CPU pelo algoritmo de [18].

Analisando os limites inferiores w_{PL} [92], dados pela Relaxação Linear de (6.1), verificamos que os limites w_d obtidos pelo algoritmo NDRC aqui introduzido também foram bons. Na maioria dos casos, estes limites ficaram bastante próximos de seus valores máximos teóricos. A exceção reside nas instâncias de 400 vértices do conjunto K. Para estas, os limites duais obtidos foram bastante distantes dos correspondentes limites de Programação Linear (em alguns casos, gaps de dualidade de cerca de 10% foram observados). Por um lado, a distância entre os limites teóricos e aqueles de fato obtidos pode ser justificada pela dificuldade do Método do Subgradiente em lidar com a simetria de custos, característica das instâncias do conjunto K. Tais simetrias podem conferir comportamento errático ou oscilatório ao método. Por outro lado, a distância entre esses limites pode também ser atribuída à dificuldade de escolher adequadamente, dentre as desigualdades GSECs, quais dualizar. Recorde (Seção 6.3.1) que adotamos a estratégia de dualizar uma única desigualdade para

cada conjunto S_i, quando, na verdade $|S_i| - 1$ desigualdades igualmente violadas induzidas por S_i existem. Outras estratégias que testamos foram:

- dualizar todas as $|S_i| - 1$ desigualdades para cada conjunto S_i;
- dualizar uma desigualdade *surrogate* obtida por uma combinação linear positiva, com mesmo peso, normalizada, das $|S_i| - 1$ desigualdades GSECs geradas a partir do conjunto S_i.

Nenhuma das alternativas acima levou a limites inferiores mais fortes que os obtidos com a estratégia inicial. Uma outra alternativa, entretanto, permitiu reduzir significativamente a distância entre w_d e w_{PL}. Esta foi a permissão de sobrevida para as desigualdades GSECs (veja Seção 6.3.2). Realizamos outras simulações para valores distitos de IDADEMAX. Os resultados destas novas simulações são apresentados na Tabela 6.12. Para cada novo valor de IDADEMAX testado, 5 ou 10, apresentamos o novo limite inferior obtido, o tempo de CPU (em segundos) e o ganho percentual oriundo da sobrevida, calculado em relação ao limite inferior obtido quando IDADEMAX = 1. Como pode ser observado, os ganhos médios nos limites inferiores para as instâncias K400 foram de 2.32% e 2.54% para IDADEMAX = 5 e 10, respectivamente. Note que estes valores fornecem uma ordem de grandeza (em valores absolutos) da queda dos gaps de dualidade apresentados para as instancias K, oriunda da sobrevida. Motivados por estes resultados positivos, submetemos todas as 38 instâncias dos conjuntos K,P,C,D e E para as quais gaps de dualidade iguais ou superiores a 1% haviam sido obtidos a uma nova simulação com SMMAX=5000, $\xi = 250$ e IDADEMAX = 5. Com estes novos parâmetros, conseguimos obter certificados de otimalidade para 4 instâncias adicionais e os gaps de dualidade médios foram reduzidos em um terço.

Tabela 6.12: Sensibilidade dos limites inferiores à sobrevida das desigualdades

	IDADEMAX= 1		IDADEMAX= 5			IDADEMAX= 10		
	w_d	t(s)	w_d	t(s)	ganho [%]	w_d	t(s)	ganho [%]
K200	316673.9170	0.74	325021.7152	0.87	2.64	323684.0345	1.16	2.21
K400	326913.4753	2.73	335354.6354	3.42	2.58	335059.1583	4.03	2.49
K400.1	448931.0463	2.75	456311.7096	4.00	1.64	463199.9214	6.18	3.18
K400.2	426979.2349	5.57	441538.5219	6.87	3.41	443030.7666	9.88	3.76
K400.3	391085.1170	2.39	401146.9746	2.78	2.57	399468.6341	3.95	2.14
K400.4	366707.8591	2.07	373036.3271	2.84	1.73	375097.5467	3.90	2.29
K400.5	495385.8162	2.33	502802.1897	3.45	1.50	505274.0306	5.33	2.00
K400.6	352497.1554	3.34	358232.1295	3.93	1.63	358363.4084	5.35	1.66
K400.7	441518.7576	3.07	451524.8872	4.44	2.27	453673.8149	7.01	2.75
K400.8	398891.5238	3.42	404459.2169	4.26	1.40	404816.8895	6.13	1.49
K400.9	357336.0506	3.00	371505.2503	4.16	3.97	371384.5287	6.44	3.93
K400.10	350741.3267	2.76	359434.9675	3.77	2.48	359876.6518	6.13	2.60

Analisando os resultados obtidos pela heurística para as instâncias H, consideradas as mais difíceis da literatura [85], verificamos que soluções primais de boa qualidade foram obtidas. Note que, para 3 das 10 instâncias H propostas em [85] e que foram submetidas ao algoritmo Branch-and-Cut naquela referência (em [85], foram divulgados resultados computacionais apenas para as instâncias H geradas para $d \leq 10$) apresentamos os melhores limites primais conhecidos em tempos de computação bastante diminutos, se comparados aos de [85] (1800 segundos de tempo máximo, em uma máquina Intel Pentium IV com 2.8 Ghz e 2 Mbytes de memória RAM).

Comparamos também nossos tempos de CPU com aqueles obtidos em [85]. Os tempos médios de CPU gastos pelo algoritmo Branch-and-Cut em [85] (associados a uma máquina Intel Pentium IV com 2.8 Ghz e 2 Mbytes de memória RAM), para resolver todas as instâncias dos conjuntos C, D, E foi de 4.9, 22.3 e 253.4 segundos, respectivamente. Comparando estes valores com os tempos médios da heurística (5.9, 34.8 e 427.8 segundos, respectivamente), verificamos que o algoritmo exato de [85] foi cerca de 1.7 vezes mais rápido que a heurística. Este resultado claramente enfatiza a qualidade da formulação do PASRP proposta em [85]. Um fato adicional que deve ser destacado é a pequena dependência do desempenho do algoritmo de [85] à qualidade do pré-processamento, para os quais nossos resultados (tanto em tempo

de CPU quanto em taxas de eliminação de variáveis) se sobressaem. Apesar desta comparação desfavorável para as instâncias dos conjuntos C,D e E, nossa heurística funcionou melhor que o algoritmo exato de [85] para as instâncias do conjunto H, na medida em que, de um modo geral, soluções primais de melhor qualidade foram obtidas, principalmente para as maiores instâncias do conjunto, em tempos de CPU de uma a duas ordens de grandeza inferiores aos de [85].

Com o objetivo de avaliar como a heurística Lagrangeana aqui proposta se compara com um algoritmo exato baseado na formulação que lhe deu origem, implementamos e testamos um algoritmo Branch-and-Cut baseado na formulação do PASRP proposta em [92]. Este assunto é tratado na Seção que segue.

6.6 Comparando Branch-and-Cut com Relax-and-Cut

O algoritmo Branch-and-Cut aqui implementado para o PASRP é baseado na formulação e no algoritmo de Planos de Corte introduzido por Lucena e Resende [92], no qual GSECs são separadas de forma exata. Nas Seções que seguem, descreveremos, brevemente, os componentes principais de nosso algoritmo exato: o gerador de cortes, o gerenciamento do cutpool, as heurísticas primais utilizadas e as demais estratégias de controle.

6.6.1 O algoritmo Branch-and-Cut

Uma Relaxação inicial de PL para o algoritmo Branch-and-Cut é obtida através de:

$$w_{PL} = \min \left\{ \sum_{e \in E} c_e x_e + \sum_{v \in V} d_v (1 - y_v) : (x, y) \in \mathcal{R} \cap (\mathbb{R}^{|E|}, \mathbb{R}^{|V|}) \right\}, \qquad (6.24)$$

onde a região poliedral $\mathcal{R}$ é definida pela interseção das desigualdades (6.2), (6.4)-(6.5) e (6.7)-(6.8). Assuma que $(\overline{x}, \overline{y})$ denota a solução ótima de (6.24). Se $(\overline{x}, \overline{y})$ não viola desigualdades GSECs, $(\overline{x}, \overline{y})$ resolve o PASRP. Caso contrário, uma ou mais GSECs podem ser introduzidas em (6.24) na forma de Planos de Corte, obtendo-se, assim, uma nova região poliedral $\mathcal{R}$ e novos limites de Relaxação Linear.

6.6.1.1 A separação de GSECs

Para resolver o problema de separação de GSECs, empregamos o algoritmo exato de Padberg e Wolsey [109]. Embora tenha sido originalmente proposto para a sepa-

ração de SECs (3.3), Lucena e Resende [92] mostraram como aquele algoritmo pode ser adaptado para também resolver, de forma exata, o problema de separação de GSECs. Este algoritmo é descrito em detalhes no Apêndice A. Cabe salientar que o problema de identificar GSECs violadas é equivalente a resolver $|V|$ problemas de fluxo máximo em uma rede capacitada convenientemente construída. Consequentemente, a separação pode ser realizada eficientemente, em tempo proporcional a $O(|V|^4)$.

Nos testes computacionais que efetuamos, verificamos ser vantajosa a introdução das $2|E|$ desigualdades GSECs induzidas por conjuntos de 2 vértices (isto é, as 2 desigualdades GSECs geradas a partir das extremidades de cada aresta de E). Estas desigualdades são armazenadas no cutpool, ao invés de serem introduzidas diretamente na matriz do PL. Assim sendo, dada a solução $(\overline{x}, \overline{y})$, primeiro investigamos a existência de GSECs violadas no cutpool. Todas elas, caso existam, são introduzidas na matriz do Programa Linear corrente e a nova Relaxação Linear é então resolvida. O algoritmo exato de [109], por sua vez, só é chamado se nenhuma desigualdade no cutpool for violada por $(\overline{x}, \overline{y})$.

Conforme mencionamos em Capítulos anteriores, uma das formas conhecidas para tentar evitar os efeitos do tailing off é o uso de cortes ortogonais [75, 92]. No caso específico do PASRP, Lucena e Resende [92] relatam reduções de tempo de CPU da ordem de 60 vezes, com sua utilização. Assim sendo, seguimos as recomendações em [92]. Em nossa implementação, a geração de cortes ortogonais é conduzida da seguinte maneira. Uma vez que o algoritmo em [109] é chamado, identificamos a desigualdade mais violada por $(\overline{x}, \overline{y})$. Caso exista, esta desigualdade é inserida no cut-pool e na matriz do subproblema. Assuma que S_{max} denota o conjunto de vértices que induz esta desigualdade de máxima violação. Em seguida, eliminamos do grafo induzido por $(\overline{x}, \overline{y})$, todas as arestas com ambas as extremidades em S_{max}. Aplicamos novamente o algoritmo em [109], tendo, agora, uma nova solução $(\overline{x}, \overline{y})$ como parâmetro de entrada. Este procedimento se repete até que nenhuma desigualdade GSEC violada seja encontrada.

6.6.1.2 Detalhes de implementação

As demais estratégias de implementação empregadas no algoritmo Branch-and-Cut são:

- Antes de iniciar o algoritmo Branch-and-Cut, aplicamos os testes de redução

descritos na Seção 6.4.1.

- Logo após a aplicação dos testes de redução, chamamos os procedimentos primais AM e BL para obter limites iniciais para o PASRP. Procuramos aprimorar estes limites chamando AM e BL ao longo da árvore de enumeração, ao final de cada nó. Isto ocorre de forma análoga à empregada na heurística Lagrangeana, isto é, empregamos *custos complementares* $\{c_e(1-\overline{x}_e) : e \in E\}$ e penalidades complementares $\{d_i(1-\overline{y}_i) : i \in V\}$ como parâmetros de entrada para o procedimento `UnrootedGrowthPhase`.

- Utilizamos o resolvedor XPRESS, versão 16.25.

- O critério de ramificação que empregamos é baseado nas variáveis. A estratégia *default* de seleção de variáveis para ramificação do software XPRESS foi empregada.

- Utilizamos o critério *Busca em Profundidade* para a seleção de nós.

- Como critério adicional de controle de tailing off, adotamos MINIMPR = 0.01% e MAXITER = 10.

- Sempre que ocorrer um acréscimo igual ou superior a MINIMPR no limite dual na última iteração de um algoritmo de Planos de Corte, eliminamos as restrições não ativas ou aquelas cuja variáveis de folga sejam básicas. Preservamos todas as desigualdades inseridas no cutpool, ao longo de toda a árvore de enumeração.

- Impusemos um tempo máximo de CPU de 3600 segundos para o algoritmo Branch-and-Cut. Uma vez que, em última análise, desejamos ter uma avaliação de como a heurística Lagrangeana se compara a um algoritmo Branch-and-Cut baseado na formulação (6.6), nos baseamos nos tempos de CPU utilizados pela heurística para estabelecer este limite de tempo. Note que para qualquer instância dos conjuntos P,K,C, D e E os tempos de CPU empregados pela heurística Lagrangena são inferiores a 1800 segundos. Apenas para as instâncias h12u e h12p, os tempos de CPU situam-se entre 1800 e 3600 segundos.

6.6.2 Comparação de resultados computacionais

Nesta Seção apresentamos e comparamos os resultados computacionais obtidos pelo algoritmo Branch-and-Cut com aqueles obtidos pela heurística Lagrangeana. Uma vez que a heurística produziu resultados melhores que aqueles obtidos pelo algoritmo exato em [85], para as instâncias H, estabelecemos a comparação para os demais conjuntos de instâncias: P, K, C, D e E .

Todos os algoritmos descritos na Seção 6.6.1 foram implementados em C. Os experimentos computacionais foram conduzidos em uma máquina AMD ATHLON de 3GHz e com 1Gbyte de memória RAM. O compilador gcc com chave de otimização `-O3` foi utilizado, no sistema operacional Unix. Acreditamos assim que os tempos de CPU gastos pelo algoritmo Branch-and-Cut podem ser comparados aos respectivos tempos de CPU da heurística Lagrangeana.

Nas Tabelas 6.13 a 6.16, apresentamos os resultados obtidos pelo algoritmo Branch-and-Cut para cada conjunto de instâncias considerado. As colunas nesta Tabela indicam, respectivamente, a instância, o melhor limite inferior global, que considera todos os nós explorados da árvore, w_G, o melhor limite primal encontrado, $\overline{w}$, o correspondente gap de dualidade, $\frac{\overline{w}-w_G}{w_G}$, em valores percentuais, o tempo de CPU em segundos e, finalmente, o número total de nós explorados pela árvore de enumeração.

Tabela 6.13: Resultados do algoritmo Branch-and-Cut, Instâncias P e K

Instância	w_G	$\overline{w}$	Gap [%]	t(s)	No. nós
P100	803300.0000	803300	Opt	0.10	1
P100.1	926238.0000	926238	Opt	0.09	1
P100.2	401641.0000	401641	Opt	0.10	1
P100.3	659644.0000	659644	Opt	0.07	1
P100.4	827419.0000	827419	Opt	0.06	1
P200	1317874.0000	1317874	Opt	0.59	1
P400	2459904.0000	2459904	Opt	3.44	1
P400.1	2808440.0000	2808440	Opt	83.09	1
P400.2	2518577.0000	2518577	Opt	2.71	1
P400.3	2951725.0000	2951725	Opt	5.88	1
P400.4	2852956.0000	2852956	Opt	44.13	1
K100	135511.0000	135511	Opt	0.11	1
K100.1	124108.0000	124108	Opt	0.14	1
K100.2	200262.0000	200262	Opt	0.19	1
K100.3	115953.0000	115953	Opt	0.47	1
K100.4	87498.0000	87498	Opt	0.05	1
K100.5	119078.0000	119078	Opt	0.11	1
K100.6	132886.0000	132886	Opt	0.05	1
K100.7	172457.0000	172457	Opt	0.06	1
K100.8	210869.0000	210869	Opt	0.20	1
K100.9	122917.0000	122917	Opt	0.04	1
K100.10	133567.0000	133567	Opt	0.07	1
K200	329211.0000	329211	Opt	1.33	1
K400	350093.0000	350093	Opt	117.57	1
K400.1	490771.0000	490771	Opt	160.90	1
K400.2	477073.0000	477073	Opt	188.84	1
K400.3	415328.0000	415328	Opt	55.04	1
K400.4	389451.0000	389451	Opt	48.09	1
K400.5	519526.0000	519526	Opt	94.43	1
K400.6	374849.0000	374849	Opt	150.98	1
K400.7	474466.0000	474466	Opt	245.59	1
K400.8	418614.0000	418614	Opt	180.44	1
K400.9	383105.0000	383105	Opt	54.00	1
K400.10	395413.0000	395413	Opt	185.94	1

Tabela 6.14: Resultados do algoritmo Branch-and-Cut, Instâncias C

Instância	w_G	$\overline{w}$	Gap [%]	t(s)	No. nós
c1A	18.0000	18	Opt	0.38	1
c1B	85.0000	85	Opt	0.58	1
c2A	50.0000	50	Opt	0.38	1
c2B	141.0000	141	Opt	0.60	1
c3A	414.0000	414	Opt	0.52	1
c3B	737.0000	737	Opt	7.19	1
c4A	618.0000	618	Opt	0.83	1
c4B	1063.0000	1063	Opt	82.02	1
c5A	1080.0000	1080	Opt	2.03	1
c5B	1528.0000	1528	Opt	33.05	1
c6A	18.0000	18	Opt	0.56	1
c6B	55.0000	55	Opt	16.63	1
c7A	50.0000	50	Opt	0.61	1
c7B	102.0000	102	Opt	1.34	1
c8A	361.0000	361	Opt	3.90	1
c8B	500.0000	500	Opt	55.39	1
c9A	533.0000	533	Opt	23.62	1
c9B	694.0000	694	Opt	415.88	1
c10A	859.0000	859	Opt	23.00	1
c10B	1069.0000	1069	Opt	23.41	1
c11A	18.0000	18	Opt	7.22	1
c11B	32.0000	32	Opt	19.91	1
c12A	38.0000	38	Opt	2.11	1
c12B	46.0000	46	Opt	44.87	1
c13A	236.0000	236	Opt	34.94	1
c13B	258.0000	258	Opt	205.15	1
c14A	293.0000	293	Opt	438.75	1
c14B	318.0000	318	Opt	538.59	1
c15A	500.6715	501	Opt	470.12	1
c15B	551.0000	551	Opt	2002.48	1
c16A	11.0000	11	Opt	40.18	1
c16B	11.0000	11	Opt	49.26	1
c17A	18.0000	18	Opt	268.11	1
c17B	18.0000	18	Opt	119.42	1
c18A	110.5000	118	6.79	3600.00	1
c18B	112.2085	118	5.17	3600.00	1
c19A	146.0000	146	Opt	8.46	1
c19B	146.0000	146	Opt	8.63	1
c20A	266.0000	266	Opt	5.34	1
c20B	267.0000	267	Opt	4.62	1

Tabela 6.15: Resultados do algoritmo Branch-and-Cut, Instâncias D

Instância	w_G	$\overline{w}$	Gap [%]	t(s)	No. nós
d1A	18.0000	18	Opt	1.59	1
d1B	106.0000	106	Opt	2.16	1
d2A	50.00000	50	Opt	1.64	1
d2B	218.0000	218	Opt	1.71	1
d3A	807.0000	807	Opt	1.91	1
d3B	1509.0000	1509	Opt	118.05	1
d4A	1203.0000	1203	Opt	5.06	1
d4B	1881.0000	1881	Opt	115.70	1
d5A	2157.0000	2157	Opt	20.62	1
d5B	3132.9963	3136	0.09	3600.00	1
d6A	18.0000	18	Opt	2.20	1
d6B	67.0000	67	Opt	87.23	1
d7A	50.0000	50	Opt	2.10	1
d7B	103.0000	103	Opt	59.99	1
d8A	755.000	755	Opt	40.20	1
d8B	1036.0000	1036	Opt	1343.00	1
d9A	1070.0000	1070	Opt	796.33	3
d9B	1418.4209	1426	0.53	3600.00	1
d10A	1671.0000	1671	Opt	1457.52	1
d10B	2077.8573	2083	0.25	3600.00	1
d11A	18.0000	18	Opt	7.72	1
d11B	29.0000	29	Opt	666.19	1
d12A	42.0000	42	Opt	77.00	1
d12B	42.0000	42	Opt	148.04	1
d13A	445.0000	445	Opt	2480.34	1
d13B	486.0000	486	Opt	1053.64	1
d14A	599.3863	607	1.27	3600.00	1
d14B	663.3465	667	0.55	3600.00	1
d15A	1037.4422	1043	0.53	3600.00	1
d15B	1105.9807	1110	0.36	3600.00	1
d16A	13.0000	13	Opt	143.05	1
d16B	12.9232	14	8.33	3600.00	1
d17A	23.0000	23	Opt	128.31	1
d17B	23.0000	23	Opt	139.19	1
d18A	216.4312	225	3.95	3600.00	1
d18B	222.5573	231	3.79	3600.00	1
d19A	305.0647	317	3.91	3600.00	1
d19B	309.1020	321	3.85	3600.00	1
d20A	529.9888	538	1.51	3600.00	1
d20B	537.0000	537	Opt	20.78	1

Tabela 6.16: Resultados do algoritmo Branch-and-Cut, Instâncias E

Instância	w_G	$\overline{w}$	Gap [%]	t(s)	No. nós
e1A	13.0000	13	Opt	13.47	1
e1B	109.0000	109	Opt	30.39	1
e2A	30.0000	30	Opt	13.66	1
e2B	170.0000	170	Opt	33.95	1
e3A	231.000	231	Opt	27.76	1
e3B	3798.0914	3841	1.13	3600.00	1
e4A	3151.0000	3151	Opt	248.37	1
e4B	3145.3770	3165	0.62	3600.00	1
e5A	5652.8586	5668	0.27	3600.00	1
e5B	7971.4277	8002	0.38	3600.00	1
e6A	19.0000	19	Opt	19.92	1
e6B	70.0000	70	Opt	295.16	1
e7A	40.0000	40	Opt	19.75	1
e7B	136.0000	136	Opt	2880.89	1
e8A	1874.3750	1908	1.79	3600.00	1
e8B	2538.3942	2609	2.78	3600.00	1
e9A	2781.0790	2799	0.64	3600.00	1
e9B	3520.1056	3573	1.50	3600.00	1
e10A	4562.2062	4597	0.76	3600.00	1
e10B	5474.3695	5508	0.61	3600.00	1
e11A	21.0000	21	Opt	230.70	1
e11B	34.0000	34	Opt	1443.21	1
e12A	49.0000	49	Opt	1886.64	1
e12B	66.0714	70	5.95	3600.00	1
e13A	1162.3343	1192	2.55	3600.00	1
e13B	1256.2098	1296	3.17	3600.00	1
e14A	1565.2133	1596	1.97	3600.00	1
e14B	1702.0033	1739	2.17	3600.00	1
e15A	2586.3414	2620	1.30	3600.00	1
e15B	2747.6183	2777	1.07	3600.00	1
e16A	13.3749	15	12.15	3600.00	1
e16B	13.3749	15	12.15	3600.00	1
e17A	22.0637	27	22.37	3600.00	1
e17B	22.0637	27	22.37	3600.00	1
e18A	547.7920	581	6.06	3600.00	1
e18B	557.9737	593	6.28	3600.00	1
e19A	737.8453	769	4.22	3600.00	1
e19B	750.0713	773	3.06	3600.00	1
e20A	1327.7833	1338	0.77	3600.00	1
e20B	1340.7833	1352	0.84	3600.00	1

Na Tabela 6.17, apresentamos uma comparação direta entre a heurística Lagrangena (NDRC) e o algoritmo Branch-and-Cut (BC). As colunas da Tabela indicam os resultados obtidos por cada algoritmo em cada conjunto de instâncias. As linhas da Tabela indicam os parâmetros de comparação utilizados. São eles, respectivamente: o número de instâncias resolvidas na otimalidade, o número de instâncias para as quais cada um dos algoritmos encontrou a solução ótima ($\overline{w} = w$), o gap de dualidade médio, em valores percentuais, o gap de dualidade médio entre as instâncias que não foram resolvidas na otimalidade, Gap NR, o tempo de médio de CPU (t), o tempo médio de CPU entre as instâncias resolvidas na otimalidade t_R e, finalmente, o tempo médio de CPU para as demais instâncias, t_{NR}. Todos os tempos de CPU são informados em segundos.

Tabela 6.17: Comparação entre a heurística Lagrangena NDRC e o algoritmo Branch-and-Cut

Critério	Conjunto de instâncias									
	P		K		C		D		E	
	NDRC	BC	NDRC	BC	NDRC	BC	NDRC	BC	NDRC	BC
Instâncias resolvidas	5	11	11	23	17	38	9	26	6	13
$\overline{w} = w$	11	11	22	23	40	38	40	37	36	15
Gap [%]	0.31	0	3.83	0	0.92	0.43	1.26	0.72	1.48	2.97
Gap NR [%]	0.62	-	7.34	-	1.59	2.63	1.62	2.41	1.74	4.41
$t(s)$	3.05	12.75	1.54	64.55	5.91	304.25	34.83	1394.53	427.78	2608.59
$t_R(s)$	0.26	12.75	0.11	64.55	0.85	130.53	5.45	332.63	32.16	392.82
$t_{NR}(s)$	5.38	-	2.85	-	9.65	3600	43.36	3600	497.60	3600

O algoritmo Branch-and-Cut resolveu, na otimalidade, 112 das 154 instâncias consideradas no estudo. Para apenas uma destas 112 instâncias (instância d9a)

mais de um nó foi aberto na árvore de enumeração. Note que para as 42 instâncias restantes, todo o tempo de CPU foi empregado no nó raiz, a despeito do critério de prevenção de tailing off que utilizamos. Assim sendo, embora os limites possíveis de se obter pela Relaxação Linear de (6.1) sejam muito fortes (veja [92] para maiores detalhes), estes resultados sugerem, pelo menos em nossa implementação, ser alto o tempo de CPU necessário para obtê-los através de um algoritmo de Planos de Corte.

Principalmente em casos como estes, é que se espera que um algoritmo NDRC seja competitivo com um algoritmo exato baseado na mesma formulação. Isto é, espera-se que o algoritmo NDRC seja capaz de obter limites inferiores próximos aos limites teóricos, em conjunto com soluções primais de boa qualidade, em tempos de CPU significativamente menores. Comparando a diferença entre os limites inferiores obtidos pela heurística Lagrangeana e pelo algoritmo Branch-and-Cut, isto é, w_d e w_G, verificamos que a heurística se comportou como o esperado. A título de exemplificação, considere as últimas 10 instâncias do conjunto E, indicadas nas Tabelas 6.10 e 6.16. Para todas estas instâncias, o limite w_d supera w_G. Note também que os tempos de CPU gastos pela heurística Lagrangeana são entre 3 e 36 vezes menores que o tempo máximo estabelecido para o algoritmo Branch-and-Cut. Recorde ainda que, para apenas cinco instâncias dos conjuntos P, K, C, D, e E a heurística Lagrangeana não encontrou a solução ótima.

Obviamente, a média dos tempos de CPU empregados pelo algoritmo Branch-and-Cut são influenciados pelo tempo máximo de 3600 segundos que lhe foi imposto. Por outro lado, uma comparação que leve em conta apenas os tempos t_R também parece ser pouco adequada. Isto porque o número de instâncias que cada algoritmo resolveu difere fortemente (112 instâncias para o BC contra 48 para a heurística) e porque a heurística resolveu, em geral, instâncias de menor dimensão que aquelas resolvidas pelo algoritmo Branch-and-Cut. Assim sendo, para efeito de avaliação da heurística Lagrangeana introduzida neste Capítulo, nos parece adequado comparar os tempos médios de CPU para a instâncias resolvidas pelo algoritmo Branch-and-Cut que não foram resolvidas pela heurística. Os resultados desta comparação são apresentados na Tabela 6.18. As duas primeiras colunas da Tabela 6.18 indicam, respectivamente, o conjunto de instâncias e o número de instâncias, em cada conjunto, que satisfazem ao critério de comparação. Nas duas colunas seguintes, apresentamos os resultados obtidos pela heurística Lagrangeana. São eles: o gap de dualidade médio, em valores percentuais e o tempo médio de CPU, em segundos. Na

Tabela 6.18: Comparação dos dois algoritmos para as instâncias resolvidas pelo algoritmo Branch-and-Cut que não foram resolvidas pela heurística Lagrangeana

Conjunto	No. de Instâncias na comparação	NDRC		BC	Razão de tempo
		gap %	tempo (s)	tempo (s)	
P	6	0.62	5.38	12.75	2.4
K	12	7.34	2.85	64.55	22.7
C	21	1.59	7.24	211.47	29.2
D	18	2.52	26.12	472.08	18.1
E	6	3.85	65.42	821.99	12.6

coluna seguinte, apresentamos o tempo médio de CPU empregado pelo algoritmo Branch-and-Cut ao resolver na otimalidade as mesmas instâncias. Na última coluna apresentamos a razão entre o tempo de CPU gasto pelo algoritmo Branch-and-Cut e o gasto pela heurística.

Como pode ser apreciado, o tempo médio de CPU gasto pela heurística para alcançar os gaps de dualidade indicados na Tabela 6.18 foi entre 2 e 29 vezes inferior ao tempo gasto pelo algoritmo Branch-and-Cut para resolver o mesmo conjunto de instâncias. Por outro lado, os gaps de dualidade apresentados naquela Tabela são predominantemente influenciados pelos limites inferiores.

Face aos argumentos apresentados acima, acreditamos que a heurística Lagrangeana introduzida neste Capítulo produziu resultados compatíveis com os esperados de uma heurística desta classe. Isto é, obteve uma boa relação de compromisso entre qualidade dos limites encontrados e tempo de CPU empregado. Acreditamos assim, que a comparação desfavorável da heurística com o algoritmo exato de [85] seja justificada pelo mérito da formulação introduzida em [85].

6.7 Comentários

Neste Capítulo, introduzimos uma heurística Lagrangena do tipo NDRC para o Problema da Árvore de Steiner com Recolha de Prêmios. A heurística emprega informação dual Lagrangena, ao longo das iterações do Método do Subgradiente, para guiar um algoritmo construtivo para o problema. Posteriormente, são aplicados procedimentos de pós processamento e de Busca Local à solução gulosa inicialmente encontrada.

Os resultados computacionais obtidos demonstram que a heurística foi capaz de encontrar soluções ótimas para a grande maioria das instâncias testadas. Para

as instâncias mais difíceis consideradas neste estudo, a heurística encontrou, em 3 casos, os melhores limites primais conhecidos. Para as demais instâncias deste conjunto, o procedimento encontrou soluções de boa qualidade em frações do tempo de CPU gasto pelo algoritmo exato de [85]. Os resultados demonstraram também que a heurística se compara favoravelmente com a Metaheurística de Canuto et al. [18].

Comparamos a heurística Lagrangena com dois algoritmos exatos: aquele introduzido em [85], baseado em uma formulação orientada para o PASRP e um algoritmo Branch-and-Cut que implementamos. Este por sua vez, é baseado na formulação (6.6) introduzida em Lucena e Resende [92]. Os resultados desta comparação indicaram que os tempos médios de CPU da heurística são superiores aos tempos médios do algoritmo de [85] para alguns conjuntos de instâncias. Entretanto, os tempos de CPU gastos pela heurística se comparam favoravelmente com os tempos de nosso algoritmo Branch-and-Cut. Desta forma, acreditamos que a diferença de desempenho, em termos de tempo de CPU, entre a heurística aqui introduzida e o algoritmo exato em [85] se justifique pelo mérito da formulação e implementação empregados em [85].

Capítulo 7

Contribuições para a Modelagem e Solução Exata do Problema de Planejamento de Extração de Madeira

7.1 Introdução

A Indústria Florestal assume papel importante em várias economias nacionais em países como Chile, Canadá, Suécia, Finlândia e Nova Zelândia [116], onde parte significativa das exportações é baseada nesta indústria. Não surpreendentemente, os problemas de planejamento envolvidos em toda a cadeia produtiva são altamente complexos, cobrindo aspectos como política de plantio, construção de estradas, política de colheita e a própria integração da produção e distribuição dos vários subprodutos da indústria.

Técnicas de Otimização e de Pesquisa Operacional têm sido aplicadas na solução de problemas de planejamento da Indústria Florestal, há pelo menos três décadas. Neste período, observou-se não apenas um aumento expressivo nas dimensões dos problemas possíveis de se resolver, mas também a extensão do uso das técnicas de solução a problemas de natureza bastante diversificada [116].

Neste Capítulo, discutimos um problema relevante para a Indústria Florestal: o Problema de Planejamento de Extração da Madeira (PPEM). Para entender o problema, assuma que uma região florestal é dividida em *células* que representam áreas de extração de madeira, isto é, áreas que possuem quantidade significativa de árvores a cortar. Nessa região, alguns pontos são previamente identificados como sendo adequados à instalação das máquinas necessárias à execução da tarefa. Essas

máquinas são necessárias para o transporte de toras de madeira dos locais de corte a outros pontos da floresta, que se conectam a uma rede de estradas. Esta, por sua vez, contém trechos que já existem fisicamente e outros a serem construídos. A rede de estradas tem por objetivo permitir que a madeira extraída alcance um ponto de saída da floresta. Desse ponto em diante, a madeira extraída será levada a centros de consumo ou de beneficiamento. No PPEM, deseja-se resolver o problema combinado de localização de maquinário e projeto da rede de estradas, maximizando o lucro associado à colheita.

Neste Capítulo, sugerimos uma nova abordagem para a solução exata do PPEM. Após adotarmos algumas hipóteses que simplificam a topologia das soluções associadas ao problema, introduzimos dois modelos com representações em grafos para o mesmo. No primeiro, o PPEM pode ser entendido como um Problema de Steiner em Grafos, sujeito à restrições adicionais. No segundo, uma variante do Problema de Steiner em Grafos com Recolha de Prêmios sujeito à restrições adicionais é proposto. Para cada um desses modelos, formulações de Programação Inteira são propostas.

Este Capítulo é organizado da seguinte forma. Na Seção 7.2, apresentamos uma revisão das metodologias propostas na literatura para resolver o problema, de forma exata ou aproximada. Para nós, é de particular interesse uma formulação de Programação Inteira Mista sugerida para o PPEM em [121] e essa será assim discutida na Seção 7.2.1. Na Seção 7.3, apresentamos algumas simplificações assumidas na nossa modelagem do problema. Os modelos em grafos que resultam destas simplificações, bem como as formulações matemáticas a eles associadas, são apresentados em seguida. Na Seção 7.4.2.2, apresentamos resultados preliminares de um algoritmo Branch-and-Cut para um dos modelos aqui propostos. Finalmente, encerramos o Capítulo na Seção 7.5, oferecendo algumas conclusões e, principalmente, possibilidades de investigação futura.

7.2 Revisão da Literatura

7.2.1 Abordagens baseadas em Programação Inteira

Pelo que conhecemos, Vera et al. [121] foram os primeiros a formular o PPEM como um Problema de Programação Inteira Mista (MIP). A partir dessa formulação, dois algoritmos de solução foram então propostos para resolver o problema. O primeiro é um algoritmo Branch-and-Bound que se utiliza do software OSL da IBM. O segundo

é uma heurística Lagrangeana.

Na primeira parte desta Seção, descrevemos a notação empregada para formular o PPEM como um MIP. Posteriormente, introduzimos as restrições do problema e sua formulação matemática como proposta em [121]. Finalmente, apresentamos e discutimos os resultados computacionais obtidos naquela referência.

Assuma que as n células da região florestal considerada são representadas pelo conjunto $C = \{1, ..., n\}$. Empregamos o termo *célula* para designar indistintamente os elementos dos seguintes conjuntos:

- M: conjunto de células a serem colhidas.
- T^k: conjunto de localidades onde o maquinário do tipo (ou tecnologia) $k \in \{1, ..., K\}$ pode ser localizado. Assumimos que não há madeira a ser colhida nessas localidades.
- S: conjunto de pontos de saída da região florestal. Estes pontos se conectam ao sistema de transporte externo e permitem que a madeira chegue a centros de consumo, fora da região florestal.
- N: conjunto dos pontos que são, potencialmente ou de fato, extremidades de trechos de estradas.

Definimos o conjunto total de pontos de localização de maquinário como $T = \bigcup_{k=1,...,K} T^k$. Para designar a *região de alcance* de um ponto de localização em T, empregamos a seguinte notação: $P_{ij}^k = 1$, se a célula $i \in M$ pode ser colhida através da célula $j \in T^k$, usando a tecnologia (ou maquinário) do tipo k, $P_{ij}^k = 0$, caso contrário. Além disto, designamos por O_i a quantidade de madeira disponível na célula $i \in M$.

A rede de estradas, por sua vez, é descrita pelo grafo $G_N = (N, E_N)$, com trechos de estradas representados pelas arestas em E_N. Alguns destes trechos, pertencentes a E_N^e, existem fisicamente, enquanto outros, pertencentes a E_N^c, poderão ser ou não construídos. Para cada trecho $(q, r) \in E_N$, o valor $K_{(q,r)}$ fornece um limite superior para a quantidade de madeira que pode ser transportado em (q, r).

Os custos utilizados na formulação são:

- α_{jk}^1: custo fixo de instalação da tecnologia k na célula $j \in T^k$;

- α_{ij}^{2k}: custo unitário de colher a célula $i \in M$ pela célula $j \in T^k$, usando-se a tecnologia k. Este custo é definido apenas se $P_{ij}^k = 1$.

- α_{qr}^3: custo de construção do segmento $(q,r) \in E_N^c$ ($\alpha_{qr}^3 = 0$, se $(q,r) \in E_N^e$).

- α_{qr}^4: custo unitário de transporte de madeira pelo segmento $(q,r) \in E_N$.

- α_s^5: custo de transporte de um ponto de saída $s \in S$ até um centro de consumo, fora da região florestal.

- δ: receita unitária proveniente da venda da madeira colhida.

As variáveis reais associadas ao modelo do problema são:

- w_{ij}^k: volume de madeira colhido da célula $i \in M$ pela célula $j \in T^k$, usando-se a tecnologia k.

- y_j: o volume total de madeira colhido a partir da célula $j \in T$.

- $f_{(q,r)}$: o fluxo de madeira no trecho de estrada (q,r).

- g_s: o volume de madeira que sai da floresta através de $s \in S$.

Em adição às variáveis reais definidas acima, o modelo também emprega as seguintes variáveis discretas:

- $x_j^k \in \{0,1\}$, que assume valor 1 se o maquinário do tipo k é instalado na célula $j \in T^k$, ou 0, em caso contrário.

- $z_{(q,r)} \in \{0,1\}$, que assume valor 1 se o trecho (q,r) é construído, ou 0, em caso contrário. Note que $z_{(q,r)} = 1, \forall (q,r) \in E_N^e$.

Apresentada a notação acima e tendo implicitamente descrito, através dela, o modelo considerado para o PPEM em [121], temos então a seguinte formulação para o

problema:

$$\max\Bigg\{ \begin{array}{rl} \delta \sum_{j\in T} y_j & - \\ \sum_{k=1}^{K} \alpha^1_{jk} x^k_j, & - \\ \sum_{k=1}^{K} \sum_{j\in T^k} \sum_{i\in M: P^k_{ij}=1} \alpha^{2k}_{ij} w^k_{ij} & - \\ \sum_{(q,r)\in E^c_N} \alpha^3_{(q,r)} z_{(q,r)} & - \\ \sum_{(q,r)\in E_N} \alpha^4_{(q,r)} f_{(q,r)} & - \\ \sum_{s\in S} \alpha^5_s g_s, & \\ \text{sujeito a (7.2) - (7.13)} & \end{array} \Bigg\}. \tag{7.1}$$

$$\sum_{k=1}^{K} \sum_{j\in T^k} P^k_{ij} w^k_{ij} \leq O_i, \ \ \forall i \in M \tag{7.2}$$

$$\sum_{k=1}^{K} \sum_{i\in M} P^k_{ij} w^k_{ij} = y_j, \ \ \forall j \in T \tag{7.3}$$

$$w^k_{ij} \leq O_i x^k_j, \ \ \forall j \in T, \forall i \in M, \forall k \in K \ : P^k_{ij} = 1. \tag{7.4}$$

$$f_{(q,r)} + f_{(r,q)} \leq K_{(q,r)} z_{(q,r)} \tag{7.5}$$

$$\sum_{(q,r)\in E_N} f_{(q,r)} - \sum_{(r,t)\in E_N} f_{(r,t)} = \begin{cases} -y_r & r \in T \\ 0 & r \in N \setminus (T \cup S) \\ g_r & r \in S \end{cases} \tag{7.6}$$

$$\sum_{s\in S} g_s = \sum_{i\in T} y_i \tag{7.7}$$

$$\sum_{k=1}^{K} x^k_j \leq 1, \ \forall j \in T \tag{7.8}$$

$$x^k_j \in \{0,1\}, \qquad \forall j \in T, \ k \in \{1,...,K\} \tag{7.9}$$

$$w^k_{ij} \geq 0, \quad \forall i \in M, \ \forall j \in T, \ k \in \{1,...,K\} \tag{7.10}$$

$$z_{(q,r)} \in \{0,1\}, \qquad \forall (q,r) \in E_n \tag{7.11}$$

$$f_{(q,r)} \geq 0, \qquad \forall (q,r) \in E_n \tag{7.12}$$

$$y_j \geq 0, \qquad \forall j \in T. \tag{7.13}$$

Antes de prosseguir, vamos nos concentrar na interpretação dos termos da função objetivo (7.1) e nas restrições (7.2)-(7.13).

O primeiro termo da função objetivo denota a receita apurada com a venda da madeira colhida. Os termos seguintes denotam custos associados à colheita e à construção de estradas. São eles, respectivamente: o custo fixo de instalação de maquinários, o custo variável de colheita, o custo fixo de construção de segmentos de estrada, o custo variável de transporte de madeira dentro da região florestal e, finalmente, o custo variável de transporte da floresta até o ponto de consumo (fora da região florestal).

As restrições (7.2) estabelecem que o volume total de madeira colhido em uma célula é limitado pela oferta de madeira naquela célula. As restrições (7.3), por sua vez, impõem que o volume total colhido por um ponto de maquinário deve igualar o volume colhido em todas as células em M que pertencem à sua região de alcance. As desigualdades (7.4) asseguram uma relação lógica: uma célula $j \in T^k$ só pode colher a madeira de uma célula $i \in M$, se esta pertencer à região de alcance de j e se a tecnologia k for instalada em j. As desigualdades (7.5) garantem que só pode haver fluxo em qualquer direção de um trecho $(q, r) \in E_N$ se o trecho for existente ou se vier a ser construído. Por outro lado, as restrições (7.6) garantem a conservação de fluxo nas extremidades dos trechos de estradas. Por sua vez, as restrições (7.7) asseguram que o volume total colhido pelas células em T deve igualar o volume total de madeira que sai da região florestal. Finalmente, as desigualdades (7.8) impõem que, no máximo, um tipo de tecnologia k pode ser instalada em cada ponto de maquinário.

Vamos tratar agora dos métodos de solução empregados em Vera et al. [121] para resolver (7.1). Em uma primeira abordagem, os autores procuraram resolver o modelo através de um algoritmo Branch-and-Bound usando, para tanto, o software OSL da IBM. Na árvore de enumeração, foi também introduzida uma heurística com objetivo de gerar soluções viáveis para o PPEM, a partir das soluções das Relaxações Lineares associadas a seus nós.

Os resultados computacionais obtidos com a primeira abordagem descrita acima demonstraram que a Relaxação Linear de (7.1)-(7.13) fornece limites inferiores fracos para o problema. Assim sendo, não se conseguiu resolver instâncias do PPEM de interesse prático, isto é, com $|M| \geq 3000$ e com grande número de trechos de estradas [121]. Posteriormente, a Relaxação Linear de (7.1)-(7.13) foi fortalecida com algumas desigualdades válidas que permitiram que uma instância um pouco mais difícil, com $|M| = 4000$ e $|N| = 65$, fosse resolvida com garantia de otimalidade. Para a instância

mais difícil do estudo ($|M| = 4000$, $|N| = 109$) gaps de dualidade da ordem de 2.6% a 7% (dependendo dos valores de δ empregados) foram obtidos, após o fortalecimento da formulação.

Face às dificuldades mencionadas, Vera et al. [121] sugeriram também uma heurística Lagrangeana baseada na relaxação das restrições (7.3). Note que se estas restrições forem relaxadas, o problema remanescente pode ser decomposto em dois subproblemas desacoplados. O primeiro deles (envolvendo as variáveis x, w) consiste em definir os pontos de localização de maquinários e as quantidades a serem colhidas, por cada um destes, em cada célula em M. A região de viabilidade deste subproblema é dada pelas restrições (7.2),(7.4) e (7.8)-(7.10). O segundo subproblema (envolvendo as variáveis f, z, y e g) consiste no projeto da rede de estradas e a região de viabilidade a ele associada é definida por (7.5)-(7.7) e (7.11)-(7.13). Explorando esta estrutura, a heurística Lagrangeana obteve gaps de dualidade entre 2.4% e 6% (após cerca de 6 horas de CPU, em uma máquina Pentium MMX de 200 Mhz e 64Mbytes) para a instância mais difícil do estudo, mencionada anteriormente.

7.2.2 Outras abordagens

Uma das ferramentas de apoio à decisão disponíveis e em uso pela indústria florestal Chilena para resolver o PPEM é o sistema PLANEX [40]. Trata-se de um sistema conectado a um módulo de gerenciamento de informações via satélite que fornece dados sobre as áreas a serem colhidas. Estas informações incluem o mapeamento da região em células de 10 metros por 10 metros, a altitude média e a disponibilidade de madeira em cada célula, o sistema de estradas de acesso já existentes e eventuais barreiras geográficas presentes na região (por exemplo rios). O sistema PLANEX se utiliza de heurísticas gulosas para a construção de soluções viáveis para o PPEM.

Uma Metaheurística do tipo Busca Tabu foi introduzida por Legües et al. [79] para resolver o PPEM. Nesse procedimento, o primeiro passo para obter uma solução viável inicial para o problema consiste em avaliar o lucro associado à colheita das células na região de alcance de cada localização de maquinário. O lucro leva em consideração a receita decorrente da venda da madeira, os custos fixos de instalação do maquinário e os custos variáveis associados à colheita. Em seguida, é criada uma lista com os pontos de localização de maquinários, ordenados em magnitude decrescente de seus lucros. A cada iteração da construção da solução inicial, os pontos de localização mais lucrativos são escolhidos. Na medida em isto ocorre, os demais

pontos em T que possuem região de alcance comum com os pontos de maquinário já escolhidos são eliminados da lista inicial. Este processo continua até que todas as células em M tenham sido alocadas a algum ponto de maquinário ou até que a lista inicial se torne vazia. A partir de então, a solução inicial é perturbada, abrindo-se (ou fechando-se) pontos de maquinário em sua vizinhança. Em função disto, ocorre também a re-atribuição de pontos de colheita a pontos de maquinário. Recorde que o custo total de cada solução para o PPEM também depende da solução do subproblema associado ao projeto da rede de estradas. Resolver este subproblema de forma ótima para cada conjunto de pontos de maquinário investigado ao longo da Busca Tabu foi considerado muito dispendioso. Assim sendo, ao invés de resolver este subproblema na otimalidade, constrói-se uma solução aproximada, dada por uma AGM que conecta os pontos de localização de maquinários escolhidos e um ponto de saída da região florestal. Ao final da Busca Tabu, as melhores soluções encontradas são submetidas a uma etapa de pós processamento. Nesta fase, o objetivo é encontrar uma rede de estradas de menor custo, a partir do conjunto de pontos de maquinário associado a cada solução. Isto é feito através de uma heurística (de boa qualidade) para um Problema de Steiner que define este problema topológico. Sob essa ótica, o conjunto de vértices *terminais* do PSG corresponde à união dos pontos de maquinários presentes em cada solução que se deseja melhorar e um ponto de saída da floresta. Os vértices não terminais, por sua vez, são os demais pontos de localização de maquinário e de saída da região florestal, assim como os pontos de interseção de segmentos de estradas. Para esta fase de pós processamento, utilizou-se o algoritmo de aproximação de Takahashi e Matsuyama [120] e, em seguida, o procedimento de Busca Tabu introduzido em [113] para o PSG.

Em [79], foram testadas instâncias com $|M| \in \{1000, 4071, 21000, 50000\}$. Para todas elas, as soluções viáveis obtidas pela heurística foram pelo menos tão boas quanto aquelas obtidas pelo software CPLEX 8.1, ao tentar resolver o modelo (7.1) através de um algoritmo Branch-and-Bound. Tomando como base os melhores limites duais obtidos ao longo da árvore de enumeração, os gaps de dualidade associados às soluções obtidas pela heurística situam-se entre 3% e 12% (dependendo dos valores de δ empregados). Os resultados obtidos em [79] foram expressivos o suficiente para determinar a inclusão da Metaheurística no sistema gerencial PLANEX.

Embora progressos substanciais em termos da qualidade das soluções obtidas e tempo de CPU empregados tenham sido obtidos em [79], a solução de grandes

instâncias do PPEM na otimalidade persiste como um grande desafio acadêmico e, principalmente, prático.

Para encerrar esta Seção, cabe resssaltar que o PPEM possui horizonte de planejamento longo e, por conseguinte, o tempo disponível para a tomada das decisões é razoavelmente grande. Assim sendo, o tempo de CPU empregado por um método de solução para o problema, embora importante, é um parâmetro secundário se comparado à qualidade das soluções obtidas por tal método. Desta forma, se viermos a dispor de uma formulação matemática realista para o PPEM, tanto física quanto economicamente, e de um algoritmo exato capaz de resolvê-la ainda que em longo espaço de tempo, poderemos contribuir com ganhos substanciais para a indústria em questão. Esta expectativa se justifica pelo fato dos custos envolvidos no problema serem enormes e, assim, pequenas reduções percentuais de custos representarem ganhos absolutos bastante significativos.

Na próxima Seção apresentamos uma modelagem do PPEM que tenta avançar nesta direção.

7.3 Modelos restritos às características essenciais do PPEM

Antes de prosseguir, vamos considerar e analisar algumas simplificações que levam a modelos do PPEM que excluem aspectos de menor relevância do problema. Estes aspectos foram indentificados em [123]. São elas:

- Não será considerada a componente de custo de transporte da floresta até o ponto de consumo. Isto significa que vamos restringir o problema à área florestal.

- Não serão consideradas as capacidades máximas de fluxo associadas aos trechos da rede de estradas. Isto pode ser feito porque, na prática, estas restrições não existem. Uma indicação disto é que, em [121], antes de tentar resolver (7.1)-(7.13), o parâmetro $K_{(q,r)}$ é estimado em função da disponibilidade de madeira na células a serem colhidas.

- Os custos variáveis (dependentes do volume colhido nas células em M e do volume transportado nos trechos de estrada) serão desconsiderados. Isto pode

ser feito porque, segundo especialistas da indústria, tais custos são muito pouco representativos quando comparados aos custos fixos de infra-estrutura.

- Assumimos que se existe colheita em uma célula $i \in M$ a partir de uma localidade $j \in T$, então toda a madeira existente em i é colhida a partir de j. Uma vez que a localidade j foi aberta, esta opção só não seria vantajosa se o custo variável de colheita fosse superior à receita de venda. Neste caso, no entanto, nenhuma fração da madeira existente na célula i deveria ser colhida. Além disso, abolimos a restrição de capacidade nos trechos de estrada.

Face às simplificações acima, vamos adotar, a partir de agora, uma notação mais conveniente para a representação matemática do PPEM. Novamente, assumimos que a floresta é particionada em células a serem colhidas. Os conjuntos M, N e T são definidos como antes. Porém, vamos considerar agora que apenas um ponto de entrada/saída $\{s\}$ existe na região florestal. Conforme poderá ser observado logo a seguir, esta imposição não implica em perda de generalidade nos nossos modelos. Assim sendo, assumimos, de agora em diante, que $S = \{s\}$.

Em virtude destas simplificações, uma solução para o PPEM passa a ter topologia de árvore. Nesta árvore, os vértices em M devem ter grau no máximo igual a um. Os vértices em $N \cup T$, por sua vez, podem ou não ser utilizados para realizar a colheita da madeira e para conectar trechos de estrada. O vértice s deve se conectar aos demais vértices do grafo por meio de arestas que envolvam vértices em $N \cup T$.

Em função da possibilidade de colher todas as células em M ou deixar de colher algumas delas, propomos dois modelos em grafos para o PPEM.

No primeiro modelo, M1, impomos que todas as células em M devem ser colhidas. Assim sendo, consideramos que uma solução do PPEM induz uma Árvore de Steiner no qual alguns vértices terminais (aqueles em M) devem possuir grau um na solução. Neste modelo, o conjunto de vértices terminais é dado por $M \cup \{s\}$. Os vértices não terminais correspondem a todos os demais vértices do grafo. Em adição aos custos das arestas, associamos custos fixos aos vértices não terminais. Naturalmente, estes custos fixos representam os custos de infra-estrutura envolvidos quando um destes vértices faz parte da solução (seja representando maquinário ou entroncamentos entre trechos de estradas). Uma vez que todas as células em M devem ser colhidas, uma solução ótima para PPEM neste modelo é uma árvore onde o custo total (isto é, a soma dos custos das arestas na árvore mais a soma das custos fixos associados

aos vértices de Steiner) é minimizado.

No segundo modelo, M2, não impomos a exigência de que a madeira existente em uma célula seja necessariamente colhida. Neste caso, consideramos que uma solução do PPEM induz uma Árvore de Steiner com Recolha de Prêmios. Em adição aos pesos das arestas, custos fixos são associados aos vértices em $T \cup N$ e receitas são associadas aos vértices em M. A função objetivo em M2 é análoga a (7.1), isto é, deseja maximizar o lucro resultante da colheita da madeira.

Uma simplificação de ordem prática adotada em [121, 79] que também adotamos aqui é a seguinte. Embora a formulação (7.1)-(7.13) nos dê flexibilidade para a instalação de uma dentre K tipos de tecnologia em uma localidade $j \in T$, $|K|$ é, em geral, muito pequeno. Frequentemente, apenas duas tecnologias são empregadas: *torres* e *skidders*. As torres, mais caras, são mais adequadas para locais íngrimes e podem transportar toras de madeiras por grandes distâncias (até 1000 metros). Os skidders, por outro lado, são utilizados em locais mais planos e seu uso é restrito a distâncias menores (até 300 metros). Em termos práticos, em função das características geográficas da região florestal, é possível saber, a priori, qual é o tipo de tecnologia mais adequada a cada localidade em T. Assim sendo, a notação T^k será então abolida. Assumimos assim que T denota o conjunto total de possíveis pontos de instalação de maquinário e que, para cada j em T, apenas uma das duas tecnologias é instalada.

Feitas estas considerações, associamos ao PPEM um grafo não direcionado $G = (V, E)$ onde $V = M \cup T \cup N \cup \{s\}$. Este grafo, com um conjunto de arestas E é tipicamente muito esparso, já que as extremidades de suas arestas representam localidades que situam-se próximas, umas das outras. Além disto, assumimos que não há aresta $e \in E$ que possua uma extremidade em M e outra em $M \cup N \cup \{s\}$, isto é, as células a serem colhidas só podem se conectar com possíveis pontos de localização de maquinário. As demais arestas de G possuem ambas as extremidades em $T \cup N \cup \{s\}$. Assim sendo, assumimos que o conjunto de arestas $E = E^1 \cup E^2$ é particionado em $E^1 = \{(i, j) : i \in M, j \in T\}$ e $E^2 = \{(i, j) : i, j \in N \cup T \cup \{s\}\}$. Note que $E^1 \cap E^2 = \emptyset$.

Os custos e receitas envolvidos nos dois modelos são os seguintes. Associamos um custo fixo $f_j \geq 0$ aos vértices em $N \cup T$. Este custo é contabilizado, por exemplo, se ocorrer a instalação de maquinário em $j \in T$. Um custo fixo $c_e \geq 0$ é atribuído a cada aresta $e = (i, j) \in E^1$. Este custo é contabilizado se toda a colheita da célula

$i \in M$ é feita através da localidade $j \in T$. Da mesma forma, atribuímos um custo fixo $d_e \geq 0$ a cada aresta $e = (i, j) \in E^2$. Este valor representa o investimento necessário para construir o trecho (i, j) da rede de estradas. Para M2, uma receita r_i é também associada aos vértices $i \in M$.

Finalmente, justificamos agora nossa afirmativa de que a imposição de um único ponto de entrada / saída s não é restritiva para M1 ou para M2. Se ao contrário de uma única entrada, um conjunto S de possíveis entradas, com $|S| > 1$, é disponível, podemos transformar o problema original nos problemas indicados por M1 e M2 da seguinte forma. Acrescentamos em G um vértice artificial 0, em conjunto com arestas artificiais $\{(0, i) : i \in S \ : \ d_{(0,i)} = 0\}$. Adicionalmente, introduzimos em N os vértices em S. Desta forma, conectar o vértice 0 às células a serem colhidas (vértices em M) só é possível através de arestas artificiais com custo nulo. Note que o vértice artificial 0 funciona exatamente como um único vértice de entrada para a floresta e que, por construção, o custo da árvore no grafo expandido iguala o custo da floresta obtida ao se remover as arestas artificiais.

Na próxima Seção, apresentamos formulações de Programação Inteira para M1 e M2.

7.3.1 Formulações de Programação Inteira para os modelos propostos

7.3.1.1 Formulações para M1

Antes de apresentarmos a primeira formulação de Programação Inteira para M1, salientamos que as variáveis de decisão empregadas aqui, x e y, assumem significado distinto daquele utilizado em (7.1)-(7.13). Assim sendo, empregamos as variáveis $\{y_i \in \{0, 1\} : i \in T \cup N\}$ para escolher os vértices de Steiner na solução. Já as variáveis $\{x_e \in \{0, 1\} : e \in E\}$ são usadas para escolher as arestas na árvore. A formulação é dada por:

$$\min w \ = \left\{ \sum_{e \in E^1} c_e x_e + \sum_{e \in E^2} d_e x_e + \sum_{j \in T \cup N} f_j y_j : (x, y) \in H_1 \cap (\mathbb{B}^{|E|}, \mathbb{B}^{|T|+|N|}) \right\}, \tag{7.14}$$

onde a região poliédrica H_1 é definida como:

$$\sum_{e \in E} x_e = \sum_{j \in T \cup N} y_j + |M|, \tag{7.15}$$

$$\sum_{e \in \delta(i)} x_e = 1, \;\; i \in M, \tag{7.16}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{j \in S \cap (T \cup N)} y_j + |S \cap (M \cup \{s\})| - 1, \;\; S \subseteq V, \;\; S \cap (M \cup \{s\}) \neq \emptyset, \tag{7.17}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{j \in S \setminus \{i\}} y_j, \;\; i \in S, \;\; S \subseteq V, \;\; S \cap (M \cup \{s\}) = \emptyset, \tag{7.18}$$

$$0 \leq x_e \leq 1, \; \forall e \in E, \tag{7.19}$$

$$0 \leq y_j \leq 1, \; \forall j \in N \cup T. \tag{7.20}$$

A formulação (7.14) é uma adaptação da formulação introduzida em [53, 87, 95] para o PSG. A diferença reside na função objetivo (aqui associamos custos aos vértices terminais) e nas restrições (7.16), que garantem que toda célula $i \in M$ é colhida através de um (único) ponto de maquinário. A restrição (7.15), por sua vez, determina o número necessário de arestas para cobrir os vértices em $M \cup \{s\}$ e os vértices de Steiner selecionados. As GSECs (7.17) e (7.18) garantem que a solução é livre de ciclos. Designamos (7.14) como a formulação GSEC de M1.

Uma formulação alternativa para M1 pode ser obtida através da adaptação da formulação de *cutsets* introduzida para o PSG por Aneja [2]. Sua estrutura poliedral foi estudada por Chopra e Rao [23, 24] e ela foi utilizada em implementações computacionais por Chopra et al. [22] e Koch e Martin [75]. A formulação alternativa, aqui denominada de formulação CUT para M1, baseia-se em um digrafo $D = (V, A)$ obtido a partir do grafo $G = (V, E)$, definido anteriormente. A construção de D a partir de G é feita da seguinte forma:

- cada aresta $e = (i, j) \in E^1$, tal que $i \in M$ e $j \in T$, dá origem a um arco $[j, i]$ em A, com custo $c_{[j,i]} = c_{(i,j)}$.
- cada aresta $e = (i, j) \in E^2$ dá origem dois arcos em A, $[i, j]$ e $[j, i]$. Para $[i, j]$, temos que $c_{[i,j]} = d_{(i,j)} + f_j$. Analogamente, para $[j, i]$, $c_{[j,i]} = d_{(i,j)} + f_i$.

Para cada arco $a \in A$, associamos uma variável de decisão $0 - 1$, h_a, que assume valor 1, caso o arco pertença a uma solução do PPEM, ou 0, caso contrário. Definimos o conjunto de arcos de A que *saem* de um conjunto de vértices $W \subset V$ como $\delta^+(W) := \{[i, j] \in A : i \in W, j \in V \setminus W\}$. Analogamente, definimos o conjunto de arcos que *chegam* a um conjunto de vértices $W \subset V$ como $\delta^-(W) := \{[i, j] \in A : i \in V \setminus W, j \in W\}$. Similarmente, denotamos por

$h(M) := \sum_{a \in M} h_a$, a soma das variáveis de decisão h associadas a um conjunto de arcos $M \subseteq A$.

Estamos agora em condição de apresentar a formulação CUT para M1 como:

$$\min w \; = \; \left\{ \sum_{a \in A} c_a h_a : h \in H_2 \cap \mathbb{B}^{|A|} \right\}, \tag{7.21}$$

onde a região poliédrica H_2 é definida por:

$$h(\delta^+(W)) \geq 1, \; \forall W \subset V, \; s \in W, \; (V \setminus W) \cap M \neq \emptyset, \tag{7.22}$$

$$h(\delta^+(i)) \begin{cases} = & 0, \quad \text{se } i = s \\ = & 1, \quad \text{se } i \in M \\ \leq & 1, \quad \text{se } i \in (T \cup N) \end{cases}, \tag{7.23}$$

$$h(\delta^-(i)) \leq h(\delta^+(i)), \forall i \in (T \cup N), \tag{7.24}$$

$$h(\delta^-(i)) \geq h_a, \forall a \in \delta^+(i), \; i \in (T \cup N), \tag{7.25}$$

$$0 \leq h_a \leq 1, \forall a \in A. \tag{7.26}$$

Essa formulação tem origem em uma reformulação proposta por Chopra e Gorres [21] para o PSG com custos positivos nos vértices. Na reformulação sugerida por Chopra e Gorres, aquele problema, originalmente definido em um grafo não orientado, passa a ser representado como um Problema de Steiner em Redes.

A Figura 7.1 indica uma solução viável para (7.22) - (7.26), onde $M = \{m_1, \ldots, m_6\}$ denota o conjunto de células a serem colhidas e $N \cup T = \{i, j, k, p, q, z\}$ denota o conjunto de vértices não terminais. Note que, se somados, os custos dos arcos indicados na Figura equivalem ao custo total de uma solução para o PPEM em G. Observe que o vértice s é a raiz da árvore de Steiner orientada que representa a solução. Observe também, que apenas um arco aponta para os demais vértices na árvore. Em particular, repare que todos os vértices de Steiner são vértices de *transbordo*.

Não é difícil perceber que os Steiner-cuts (7.22) garantem haver um caminho orientado entre s e qualquer conjunto W que contenha um vértice terminal, no grafo induzido por uma solução do PPEM em D. Note também que as restrições (7.23) asseguram que qualquer vértice distinto da raiz só pode ter grau *entrante* igual a um, se for um vértice terminal, ou, no máximo igual a um, se for um vértice não terminal. As desigualdades (7.24) e (7.25) estabelecem que um vértice de Steiner é necessariamente um *vértice de passagem*. Isto se justifica uma vez que $\{c_a \geq 0 : a \in A\}$.

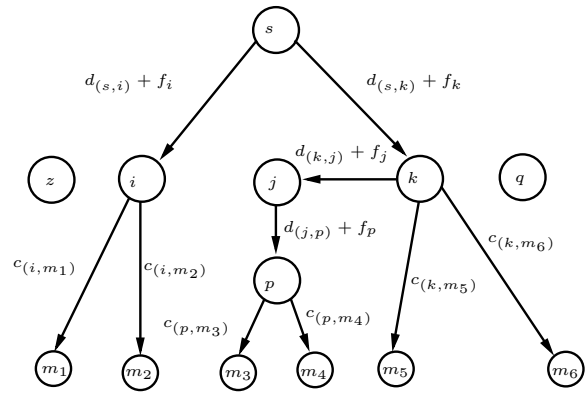

Figura 7.1: Uma solução no digrafo D para o modelo M1 de PPEM

7.3.1.2 Uma formulação para M2

Similarmente à formulação GSEC para M1, uma formulação GSEC para M2 envolve dois conjuntos de variáveis: $\{x_e \in \{0,1\} : e \in E\}$ e $\{y_i \in \{0,1\} : i \in V\}$. As variáveis no primeiro conjunto definem a topologia da árvore resultante. As variáveis do segundo conjunto, por outro lado, são usadas para definir as células a serem colhidas, os pontos onde serão instalados maquinário e os cruzamentos de estradas. Uma formulação para M2 é dada por:

$$\text{maximize } l = \left\{ \sum_{i \in M} r_i y_i - \sum_{e \in E^1} c_e x_e - \sum_{e \in E^2} d_e x_e - \sum_{j \in T \cup N} f_j y_j : (x,y) \in H_3 \cap (\mathbb{B}^{|E|}, \mathbb{B}^{|V|}) \right\} \tag{7.27}$$

onde H_3 é a região poliédrica definida por:

$$\sum_{e \in E} x_e = \sum_{j \in V} y_j - 1, \tag{7.28}$$

$$\sum_{e \in \delta(i)} x_e = y_i, \;\; i \in M, \tag{7.29}$$

$$\sum_{e \in \delta(s)} x_e \geq y_s, \tag{7.30}$$

$$y_s = 1, \tag{7.31}$$

$$\sum_{e \in \delta(i)} x_e \geq 2y_i, \;\; i \in N \cup T, \tag{7.32}$$

$$\sum_{e \in E(S)} x_e \leq \sum_{i \in S \setminus \{j\}} y_i, \;\; j \in S, \;\; S \subseteq V, \tag{7.33}$$

$$0 \leq x_e \leq 1, \; \forall e \in E, \tag{7.34}$$

$$0 \leq y_i \leq 1, \; \forall i \in V \setminus \{s\}. \tag{7.35}$$

Observe que a formulação (7.27) é baseada na formulação (6.6) para o PASRP, discutida no Capítulo 6. A diferença entre as duas está na função objetivo e nas restrições (7.30) que, no contexto do PPEM, assumem a forma de igualdade.

7.4 Um algoritmo Branch-and-cut para a formulação GSEC de M1

Nesta Seção, descrevemos um algoritmo Branch-and-Cut que implementamos para resolver (7.14). O algoritmo é similar àquele que implementamos para o PASRP (veja Seção 6.6.1), uma vez que ambos baseiam-se na separação exata de GSECs.

Uma Relaxação inicial de PL para o algoritmo Branch-and-Cut é dada por:

$$w_{PL} = \min \left\{ \sum_{e \in E^1} c_e x_e + \sum_{e \in E^2} d_e x_e + \sum_{j \in T \cup N} f_j y_j : (x, y) \in H \cap (\mathbb{R}^{|E|}, \mathbb{R}^{|N \cup T|}) \right\}, \tag{7.36}$$

onde a região poliedral H é definida pela interseção das desigualdades (7.15), (7.16), (7.19) e (7.20). Assuma que $(\overline{x}, \overline{y})$ denota a solução ótima de (7.36). Se $(\overline{x}, \overline{y})$ não viola desigualdades GSECs (7.17) e (7.18), $(\overline{x}, \overline{y})$ resolve PPEM. Caso contrário, uma ou mais GSECs podem ser introduzidas em (7.36) na forma de Planos de Corte, obtendo-se, assim, uma nova região poliedral H e novos limites de Relaxação Linear. Para resolver o problema de separação de GSECs, empregamos o algoritmo exato de Padberg e Wolsey [109], descrito no Apêndice A. Assim como no caso do PASRP, verificamos ser vantajosa a introdução das GSECs induzidas pelas arestas de E no cutpool. Assim sendo, dada a solução $(\overline{x}, \overline{y})$, primeiro investigamos a existência de GSECs violadas no cutpool. Todas elas, caso existam, são introduzidas na matriz do Programa Linear corrente e a nova Relaxação Linear é então resolvida. O algoritmo exato de [109], por sua vez, só é chamado se nenhuma desigualdade no cutpool for violada por $(\overline{x}, \overline{y})$. Nos algoritmos de Planos de Corte, empregamos cortes ortogonais conforme descritos na Seção 6.6.1.

Utilizamos o software XPRESS, versão 16.25. Os critérios de busca em profundidade e ramificação baseado em variáveis foram empregados em nossa implementação. Impusemos um tempo máximo de CPU de 10 horas para a execução do algoritmo.

7.4.1 Heurísticas primais

Para a obtenção de soluções viáveis iniciais para o PPEM, implementamos duas heurísticas. A primeira delas, CM, é baseada em *caminhos mínimos* envolvendo os vértices terminais de G. Esta heurística é bastante simples. Primeiro, encontramos um caminho mínimo entre s e um vértice terminal, digamos i. Naturalmente, os vértices internos a este caminho devem pertencer a $T \cup N$. Continuamos determinando caminhos mínimos entre s e os demais vértices terminais, por exemplo $j \neq i$. Este procedimento se repete enquanto houver algum vértice terminal ainda não conectado. Ao final, a solução obtida é uma árvore onde todos os vértices terminais são folhas.

A segunda heurística, TMA, é baseada em uma adaptação do algoritmo de Takahashi e Matsuyama [120] (TM) para as restrições (7.16). A adaptação de TM que resulta em TMA é bastante simples. Basta mantermos a distância entre os vértices terminais em M e os demais vértices do grafo, distintos de s, sem atualização (isto é, mantemos estes valores em ∞), ao longo de todo o algoritmo.

7.4.2 Experimentos computacionais

7.4.2.1 Conjuntos de problemas testados

Nos experimentos computacionais que conduzimos, 3 instâncias foram geradas para cada valor de $|M|$ em $\{1600, 3600, 6400\}$. As demais dimensões das instâncias testadas são indicadas na Tabela 7.1. Os nomes das mesmas, indicados na primeira coluna da Tabela, são claramente indentificados pelo valor do $|M|$ correspondente, seguido por um indexador da instância.

Dentro do possível, empregamos dados da indústria para gerar as instâncias consideradas neste estudo. Os custos de operação e os critérios estabelecidos para definir as regiões de alcance foram similares aos observados na prática. Por outro lado, o modelo geográfico adotado para a floresta é um pouco menos elaborado. A proporção de vértices observados para $|T|$ e $|N|$ em relação a $|M|$ são comparáveis àquelas observadas para as instâncias mais difíceis em [79].

Assumimos que a região florestal a ser colhida corresponde a um quadrado com $10\sqrt{|M|}$ metros de largura. Por exemplo, se $|M| = 1600$, a região florestal corresponde a uma área representada por um quadrado de lado igual a 400 metros. Isto significa que a área de cada célula a ser colhida em nosso modelo é exatamente igual

Tabela 7.1: Dimensões das instâncias

Instância	$\|M\|$	$\|T\|$	$\|N\|$	$\|V\|$	$\|E_1\|$	$\|E_2\|$	$\|E\|$
1600_1	1600	24	8	1633	26320	528	26848
1600_2	1600	24	8	1633	28200	528	28728
1600_3	1600	24	8	1633	26680	528	27208
3600_1	3600	54	18	3673	113280	2628	115908
3600_2	3600	54	18	3673	110940	2628	113568
3600_3	3600	54	18	3673	107400	2628	110028
6400_1	6400	96	32	6529	254560	8256	262816
6400_2	6400	96	32	6529	269920	8256	278176
6400_3	6400	96	32	6529	310880	8256	319136

à área das células informadas pelo sistema PLANEX (100m^2).

Consideramos que o ponto de saída s situa-se no centro da base inferior do quadrado que representa a área florestal.

Cada célula em M é representada por seu centro de massa, isto é, o centro de um quadrado de 10 metros por 10 metros de largura. Cada vértice $i \in M$ corresponde a um destes pontos do plano Euclidiano. Os vértices em T e N também correspondem a um ponto no quadrado de lado $10\sqrt{|M|}$. As coordenadas inteiras de cada um são sorteadas no intervalo $[1, 10\sqrt{|M|}]$ com probabilidade uniforme. Para cada instância, adotamos $|T| = 0.015M$. Os vértices em N são gerados da mesma forma, adotando $|N| = 0.005M$.

A tipo de tecnologia associado a cada vértice $j \in T$ também é sorteado. A probabilidade de que uma torre seja instalada em um ponto de maquinário é de $\frac{1}{3}$. Assim sendo, a probabilidade associada ao sorteio de um skidder é de $\frac{2}{3}$. O custo fixo associado à construção de uma torre é de 3500, enquanto que o custo fixo necessário para instalação de um skidder é de 100. Assumimos não haver custos fixos associados aos vértices em N.

No modelo que empregamos, o subgrafo de G, induzido pelos vértices em $\{s\} \cup T \cup N$, é completo, isto é, para todo par de vértices distintos $i, j \in (\{s\} \cup T \cup N)$, existe o correspondente segmento de estrada (i, j) em E^2. Os custos de construção destes trechos é dado pela expressão $\lfloor 6.7 f \text{dist(i,j)} \rfloor$, onde f é um número aleatório uniformemente gerado no intervalo $[0.85, 1.15]$ e dist(i,j) denota a distância Euclidi-

ana (em metros) entre os pontos i e j.

Para cada par de vértices $i \in M, j \in T$, geramos as arestas em E^1 da seguinte forma. Se a tecnologia instalada em j é uma torre e dist(i,j) é igual ou inferior a 1000 metros, a aresta (i, j) faz parte do grafo. Caso contrário, i não pertence à região de alcance de j. Se, por outro lado, a tecnologia instalada em j é um skidder, a aresta (i, j) só existe se dist(i,j) é igual ou inferior a 300 metros. O custo de uma aresta $(i, j) \in E^1$, isto é, o custo de colher uma célula i por uma célula j é sorteado no intervalo $[40, 100]$ com probabilidade uniforme.

7.4.2.2 Resultados computacionais preliminares

O algoritmos descritos aqui foram implementados em C. Utilizamos o compilador gcc (com chave de otimização -O3 ativada) no sistema operacional Linux. Os resultados que divulgamos nesta Seção foram obtidos em uma máquina com processador AMD ATHLON 2000 com 1Gbyte de memória RAM.

Na Tabela 7.2, apresentamos resultados preliminares de nosso algoritmo Branch-and-Cut para o modelo M1 de PPEM. A primeira coluna naquela Tabela indica a instância considerada. Nas duas colunas seguintes, apresentamos limites superiores obtidos pelas heurísticas CM e TMA, respectivamente. Nas colunas restantes, indicamos os resultados obtidos pelo algoritmo Branch-and-Cut. Estas colunas indicam, respectivamente, o limite de Programação Linear, w_{PL}, o melhor limite dual global obtido (considerando todos os nós abertos na árvore), w_G, o melhor limite superior encontrado, $\overline{w}$, o gap de dualidade correspondente, $\frac{\overline{w}-w_G}{w_G}$, em valores percentuais, o tempo de CPU empregado (em segundos), e, finalmente, o número de nós abertos na árvore de enumeração.

Como pode ser observado, as soluções obtidas pelas heurísticas CM e TMA não apresentam boa qualidade. No entanto, é de se esperar melhores resultados caso as mesmas sejam chamadas repetidas vezes, sob custos perturbados (vide Seções 4.2.2 e 6.3.3.1). Da mesma forma, o uso de Otimização Local deve impactar favoravelmente nos resultados. Em relação às melhores soluções encontradas ao longo da árvore de enumeração, os gaps observados para as heurísticas CM e TMA foram de 43.6% e 47.3%, respectivamente. De certa forma, estes resultados nos surpreenderam. Isto porque, para o PSG, o fator de aproximação de TM é bem inferior ao fator de aproximação de CM. Além disto, os resultados da literatura indicam que, na prática, a heurística TM fornece soluções de boa qualidade para o PSG. Uma possível

explicação para os resultados obtidos é o fato de que as heurísticas CM e TMA não consideram nenhuma informação relativa aos custos fixos associados aos vértices, para a construção das soluções.

Embora naturalmente pudessemos ter empregado as heurísticas ao longo da árvore Branch-and-Bound sob custos perturbados, em uma primeira abordagem, não o fizemos. Para os resultados preliminares que apresentamos aqui, procuramos avaliar sobretudo a qualidade dos limites inferiores e o quão boas as primeiras soluções viáveis para o PPEM são obtidas, através do critério de Busca em Profundidade. Assim sendo, os limites $\overline{w}$ indicados na Tabela 7.2, referem-se às melhores soluções viáveis encontradas pelo algoritmo de Planos de Corte sozinho, considerando todos os nós abertos na árvore.

O algoritmo Branch-and-Cut, por sua vez, resolveu 7 das 9 instâncias do estudo. Cinco destas instâncias foram resolvidas no nó raiz, sem a necessidade de ramificação. Para as demais instâncias resolvidas, poucos nós foram abertos (no máximo 3 nós). Para as duas instâncias não resolvidas na otimalidade, o gap de dualidade médio foi de 0.73%. Estes resultados indicam que bons limites primais foram obtidos logo nos primeiros nós da árvore e que os limites inferiores dados por w_{PL}, para o conjunto de instâncias testado, são fortes.

As instâncias consideradas neste estudo diferem daquelas encontradas na literatura [121, 79]. Assim sendo, a comparação de nossos resultados computacionais com as de outros autores fica comprometida. Até o momento, não conseguimos transformar as instâncias usadas em [79] em instâncias correspondentes para o modelo M1 que empregamos.

Acreditamos que os experimentos computacionais que conduzimos, embora restritos, nos permitam fazer algumas conjecturas, uma vez que as instâncias que empregamos foram geradas através de parâmetros observados na Indústria Florestal. Na medida em que as hipóteses que assumimos para o modelo sejam realistas e que os limites w_{PL} sejam fortes, algoritmos exatos do tipo Branch-and-Cut baseados nos modelos que propomos devem ser competitivos com o uso de um software MIP para resolver o modelo (7.1)-(7.13).

Acreditamos também que melhores tempos de CPU que os apresentados nesta Seção poderão ser alcançados, por exemplo, através de um algoritmo Branch-and-Cut para a formulação CUT de M1. No caso do PSG, resultados da literatura [91] indicam que a formulação baseada em cutsets leva a algoritmos Branch-and-Cut

Tabela 7.2: Resultados computacionais preliminares - Formulação de Steiner para o PPEM

Instância	Limites primais		Algoritmo Branch-and-Cut					
	CM	TMA	w_{PL}	w_G	$\overline{w}$	gap %	t(s)	Nós
1600_1	111701	111701	77753.50	77781.00	77781	Opt	29.97	3
1600_2	110765	110765	76869.00	76869.00	76869	Opt	18.37	1
1600_3	111788	111788	81692.00	81692.00	81692	Opt	56.56	1
3600_1	240130	252803	170706.00	170706.00	170706	Opt	467.64	1
3600_2	253710	249969	169242.00	169242.00	169242	Opt	480.39	1
3600_3	237388	254947	172627.00	172627.00	172627	Opt	751.03	1
6400_1	447104	449118	304378.324	304379.673	307698	1.09	36000.00	5
6400_2	448383	448383	302982.145	302983.404	304065	0.36	36000.00	7
6400_3	414299	448198	293941.000	294007.000	294007	Opt	18004.30	3

mais rápidos que aqueles obtidos para a formulação GSEC. É de se esperar que, a exemplo do PSG, os limites duais gerados pelas duas formulações, M1 e M2, sejam idênticos [53].

Assim sendo, satisfeitas estas condições, a abordagem que apresentamos aqui poderá contribuir para a resolução, na otimalidade, de instâncias maiores que aquelas resolvidas na literatura até o presente momento.

7.5 Comentários finais e perspectivas

Neste Capítulo, estudamos o Problema de Planejamento de Extração de Madeira. Após adotarmos algumas hipóteses que simplificam a topologia das soluções associadas ao problema, introduzimos dois modelos com representações em grafos para o mesmo. Para cada modelo, apresentamos formulações de Programação Inteira.

Implementamos um algoritmo Branch-and-Cut para um dos modelos propostos para o problema. Embora limitados, os resultados computacionais obtidos por este algoritmo indicam que os limites inferiores dados por uma das formulações propostas são fortes. Algumas instâncias do estudo foram resolvidas no nó raiz da árvore de enumeração.

Estes resultados computacionais, preliminares, é bem verdade, sugerem que o refino deste algoritmo, bem como a implementação de outros algoritmos Branch-and-Cut para formulações matemáticas mais eficientes para o mesmo modelo em grafos,

talvez permitam resolver na otimalidade problemas reais de dimensões superiores às atualmente resolvidas.

Para que alcancemos estes objetivos, consideramos ser importante investir nos seguintes aspectos:

- Refinamento das heurísticas propostas. Isto poderá ser feito, por exemplo, através da implementação de um procedimento de Otimização Local nos moldes daquele descrito na Seção 6.3.3.2. Os testes de dominância lá descritos podem ser adaptados para o problema aqui tratado. O uso de custos complementares ao longo da árvore de enumeração é outra alternativa que poderá melhorar a qualidade das soluções obtidas pelas nossas heurísticas.

- Implementar um algoritmo Branch-and-Cut baseado na formulação cutsets. Resultados da literatura [91] indicam que algoritmos Branch-and-Cut para formulações cutsets do PSG são mais rápidos que os correspondentes algoritmos baseados na formulação GSEC. Esperamos que estes algoritmos se comparem de forma semelhante no caso do problema aqui tratado.

- Desenvolver heurísticas para a separação de GSECs. Tais heurísticas poderão tornar mais rápidos o algoritmo Branch-and-Cut aqui apresentado, bem como aquele que pretendemos implementar para o segundo modelo proposto.

Capítulo 8

Conclusões

Nesta Tese, apresentamos três trabalhos que tratam da solução de problemas que envolvem a determinação de árvores ótimas em grafos.

Na primeira parte da Tese, tratamos do Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices, PARG. Inicialmente, estudamos o problema do ponto de vista poliedral. Mostramos que as desigualdades Blossom [35], Combs [58] e Clique Trees [60] são válidas e fortalecem uma formulação de Programação Inteira Clássica para o PARG. Ao final do estudo teórico, apresentamos algumas condições sob as quais as desigualdades da formulação clássica e aquelas que introduzimos geram facetas do politopo do k-PARG, um caso particular do PARG.

Em uma etapa posterior do estudo do PARG, nos concentramos em propor métodos de solução para o problema. Pela primeira vez, em seus 25 anos de história, fomos capazes de fortalecer, pela dualização de desigualdades Blossom em um algoritmo Lagrangeano do tipo Relax-and-Cut, os limites inferiores dados pela formulação clássica do PARG. Os resultados computacionais que obtivemos com este algoritmo indicam que nossos limites inferiores (que usam desigualdades Blossom) são mais fortes que os limites Lagrangeanos e de Programação Linear obtidos na literatura [15]. Os resultados indicam ainda que nossos limites superiores são competitivos com aqueles obtidos pelas melhores heurísticas conhecidas para o problema [114, 1].

Uma contribuição adicional que apresentamos foi um procedimento para tentar caracterizar os hiperplanos cuja interseção define um dado ponto extremo do politopo das árvores geradoras. Este procedimento permite transportar nossos limites duais Lagrangeanos diretamente para um algoritmo de Planos de Corte, de forma extremamente conveniente, não exigindo a resolução de qualquer problema de se-

paração. Apesar de ter sido proposto no contexto do PARG, este procedimento pode ser estendido, de forma direta, para outros problemas que admitem relaxações definidas sobre matróides.

Encerramos o nosso estudo do PARG com a introdução de um algoritmo híbrido Relax-and-Cut / Branch-and-Cut para a resolução exata do problema. Os resultados computacionais que obtivemos demonstram que o uso de desigualdades Blossom e o uso do procedimento para tentar caracterizar um ponto extremo do politopo das árvores geradoras contribuiu decisivamente para o melhor desempenho do algoritmo aqui proposto. Futuramente, poderemos aprimorar este algoritmo através da implementação e uso de heurísticas de separação para desigualdades Combs e Clique Trees. O algoritmo Branch-and-Cut poderá se beneficiar também pela utilização de geração de colunas em conjunto com a geração de cortes. Assim sendo, os problemas de Programação Linear a se considerar irão envolver um pequeno número de váriaveis (arestas), ou seja, apenas aquelas variáveis com custos reduzidos não negativos.

Na segunda parte da Tese, tratamos do Problema de Steiner em Grafos com Recolha de Prêmios (PASRP). Para resolver este problema, implementamos um algoritmo Relax-and-Cut, onde o método de Relaxação Lagrangeana é aplicado à formulação do PASRP introduzida em [92]. À primeira vista, esta formulação não parece adequada à utilização de Relaxação Lagrangeana. Procedemos então a uma reformulação da mesma, introduzindo um novo conjunto de variáveis que confere ao PASRP uma interpretação em grafos mais conveniente. Sob essa reformulação, o problema torna-se bastante atraente para ser decomposto de forma Lagrangena. Finalmente, concluindo o estudo, implementamos um algoritmo Branch-and-Cut baseado na formulação do PASRP introduzida em [92] e o comparamos com a heurística Lagrangeana aqui sugerida. Os resultados desta comparação indicam que a heurística Lagrangeana aqui introduzida produziu resultados computacionais compatíveis com aqueles esperados por um algoritmo desta classe: uma boa relação de compromisso entre a qualidade das soluções encontradas e o tempo de CPU empregado. Futuramente, pretendemos avaliar a utilização de outros métodos de solução (por exemplo, o Método de Bundle e o Algoritmo do Volume) para o Problema Dual Lagrangeano associado à relaxação que apresentamos para o problema.

Na última parte da Tese, sugerimos uma nova abordagem para a solução exata do Problema de Planejamento de Extração de Madeira. Após adotarmos algumas

hipóteses que simplificam a topologia das soluções associadas ao problema, introduzimos dois modelos com representações em grafos para o mesmo. Apresentamos formulações matemáticas para estes modelos e implementamos um algoritmo Branch-and-Cut para uma destas formulações. Embora limitados, os resultados computacionais obtidos sugerem que o aprimoramento do algoritmo proposto, bem como o desenvolvimento de novos algoritmos baseados nas formulações introduzidas, poderão permitir que instâncias de interesse prático sejam resolvidas no futuro.

Referências Bibliográficas

[1] Andrade, R., Lucena, A., e Maculan, N. "Using Lagrangian dual information to generate degree constrained spanning trees". *Discrete Applied Mathematics 154(5)* (2006), 703–717.

[2] Aneja, Y. "An integer linear programming approach to the Steiner problem in graphs". *Networks 10* (1980), 167–178.

[3] Bahiense, L., Maculan, N., e Sagastizábal, C. "The volume algorithm revisited: relation with bundle methods". *Mathematical Programming 94* (2002), 41–70.

[4] Balas, E. "The prize collecting traveling salesman problem". In *ORSA / TIMS Meeting in Los Angeles* (1986).

[5] Balas, E., e Christofides, N. "A restricted Lagrangian approach to the traveling salesman problem". *Mathematical Programming 21* (1981), 19–46.

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

[7] Bazaraa, M. S., e Sherali, H. D. "On the choice of step size in subgradient optimization". *European Journal of Operational Research 7* (1981), 380–388.

[8] Bazarra, M. S., e Goode, J. J. "A survey of various tactics for generating multipliers in the context of Lagrangian duality". *European Journal of Operational Research 3* (1979), 322–338.

[9] Beasley, J. "An SST-Based Algorithm for the Steiner Problem in Graphs". *Networks 19* (1989), 1–16.

[10] Beasley, J. "Lagrangean Relaxation". In *Modern Heuristic Techniques*, C. Reeves, Ed. Blackwell Scientific Press, Oxford, 1993.

[11] Bellman, R. *Dynamic Programming.* Princeton University Press, 1957.

[12] Belloni, A., e Lucena, A. "A Relax and Cut Algorithm for the Traveling Salesman Problem". In *17th International Symposium on Mathematical Programming* (2000).

[13] Belloni, A., e Lucena, A. "A Lagrangian Heuristic for the Linear Ordering Problem". In *Metaheuristics: Computer Decision-Making*, M. Resende e J. P. de Sousa, Eds. Kluwer Academic, Norwell, MA, 2003, pp. 123–151.

[14] Belloni, A., e Sagastizábal, C. Dynamic Bundle Methods. Relatório Técnico A 2004/296, Instituto de Matemática Pura e Aplicada, IMPA, Rio de Janeiro, 2004.

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

[16] Calheiros, F., Lucena, A., e de Souza, C. "Optimal Rectangular Partitions". *Networks 41* (2003), 51–67.

[17] Camerini, P., Fratta, L., e Maffioli, F. "On improving relaxation methods by modified gradient techniques". *Mathematical Programming Study 3* (1975), 26–34.

[18] Canuto, S., Resende, M., e Ribeiro, C. "Local search with perturbations for the prize collecting Steiner tree problem in graphs". *Networks 38* (2001), 50–58.

[19] Caprara, A., e Fischetti, M. "$\{0, \frac{1}{2}\}$ -Chvátal-Gomory cuts". *Mathematical Programming 74* (1996), 221–235.

[20] Carr, R. "Separating clique trees and bipartion inequalities having a fixed number of handles and teeth in polinomial time". *Math. Oper. Res. 22* (1997), 257–265.

[21] Chopra, S., e Gorres, E. On the node weighted Steiner Tree problem. Relatório técnico, New York University, New York, 1991.

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

[23] Chopra, S., e Rao, M. R. "The Steiner tree problem I: Formulations, compositions and extension of facets". *Mathematical Programming 64* (1994), 209–230.

[24] Chopra, S., e Rao, M. R. "The Steiner tree problem II: properties and classes of facets". *Mathematical Programming 64* (1994), 231–246.

[25] Craig, G., Krishnamoorthy, M., e Palaniswami, M. "Comparison of heuristic algorithms for the degree constrained minimum spanning tree". In *Metaheuristics: Theory and Applications*, I. Osman e J. Kelly, Eds. Kluwer, Boston, 1996.

[26] Croes, G. A. "A Method for Solving Traveling Salesman Problems". *Operations Research 6* (1958), 791–812.

[27] Crowder, H., e Padberg, M. "Solving large-scale symmetric travelling salesman problems to optimality". *Management Science 26* (1980), 495–509.

[28] Cunha, A., e Lucena, A. "Lower and Upper Bounds for the Degree-Constrained Minimal Spanning Tree Problem". *Networks* (2005). In press.

[29] Dantzig, G. "Discrete-Variable function Problems". *Operations Research 5* (1966), 266–277.

[30] Dantzig, G., Fulkerson, D., e Johnson, S. "Solution of a large scale traveling salesman problem". *Operations Research 2* (1954), 393–410.

[31] D.Bienstock, Goemans, M., Simchi-Levi, D., e Williamson, D. "A note on the prize collecting travelling salesman problem". *Mathematical Programming 59* (1993), 413–420.

[32] de Moraes Palmeira, M., Lucena, A., e Porto, O. A relax and cut algorithm for the quadratic knapsack problem. Relatório técnico, Departamento de Administração, Universidade Federal do Rio de Janeiro, 1999.

[33] Duin, C. *Steiner's Problem in Graphs.* PhD thesis, University of Amsterdam, 1993.

[34] Duin, C., e Voss, S. "Efficient Path and Vertex Exchange in Steiner Tree Algorithms". *Networks 29* (1997), 89–105.

[35] Edmonds, J. "Maximum Matching and a Polyhedron with 0-1 Vertices". *J. Res. Nat. Bur. Standards 69B* (1965), 125–130.

[36] Edmonds, J. "Matroids and the Greedy Algorithm". *Mathematical Programming 1* (1971), 127–136.

[37] Edmonds, J., e Johnson, E. "Matching: A Well-Solved Class of Integer Linear Programs". In *Proceedings of the Calgary International Conference on Combinatorial Structures and their Applications* (1970), R. G. et al., Ed., Gordon and Breach, pp. 89–92.

[38] Elf, M., Gutwenger, C., Jünger, M., e Rinaldi, G. "Branch and cut algorithms for Combinatorial Optimization and their Implementation in ABACUS". In *Computational Combinatorial Optimization*, M. Jünger e D. Naddef, Eds. Springer, Berlin, 2001, pp. 157–222.

[39] Engewall, S., M.Gothe-Lundgren, e Varbrand, P. "A Strong Lower Bound for the Node Weighted Steiner Tree Problem". *Networks 31* (1998), 11–17.

[40] Epstein, R., Morales, R., Serón, J., e Weintraub, A. "Use of OR Systems in the Chilean Forest Industries". *Interfaces 29: 1* (1999), 7–29.

[41] Escudero, L., Guignard, M., e Malik, K. "A Lagrangian relax and cut approach for the sequential ordering with precedence constraints". *Annals of Operations Research 50* (1994), 219–237.

[42] Fischer, M. L. "The Lagrangian relaxation method for solving integer programming problems". *Management Science 27(1)* (1981), 1–18.

[43] FUMERO, F. "A modified subgradient algorithm for Lagrangian relaxation". *Computers and Operations Research 28* (2001), 33–52.

[44] GABOW, H. "A good algorithm for smallest trees with a degree constraint". *SIAM Journal on Computing 6* (1977), 139–150.

[45] GAMBLE, A., E PULLEYBLANK, W. "Forest covers and a polyhedral intersection theorem". *Mathematical Programming 45* (1989), 49–58.

[46] GAREY, M., E JOHNSON, D. *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, New York, 1979.

[47] GAVISH, B. "Topological design of Centralized Computer Networks - formulations and algorithms". *Networks 12* (1982), 355–377.

[48] GAVISH, B. "Augmented Lagrangean Based Algorithms for Centralized Network design". *IEEE Trans. on Communications 33* (1985), 1247–1257.

[49] GEOFFRION, A. M. "Lagrangian Relaxation for integer programming". *Mathematical Programming Study 2* (1974), 82–114.

[50] GERARDS, A. M. H. "Matching". In *Handbooks in OR and MS*, O. B. et al, Ed., vol. 7. North-Holland, Amsterdam, 1995, pp. 135–224.

[51] GLOVER, F., E KLINGMAN, D. "Finding minimum spanning trees with a fixed number of links at a node". In *Combinatorial Programming: methods and applications*, B. Roy, Ed. D. Reidel Publishing Company, Dordrecht-Holland, 1975, pp. 191–201.

[52] GLOVER, F., E KLINGMAN, D. "Recent developments in computer implementation technology for network flow algorithms". *INFOR 20* (1982), 433–452.

[53] GOEMANS, M. "The Steiner tree polytope and related polyhedra". *Mathematical Programming 63* (1994), 157–182.

[54] GOEMANS, M., E WILLIAMSON, D. "The primal dual method for approximation algorithms and its applications to network design problems". In *Approximation algorithms for NP-hard problems*, D. S. Houchbaum, Ed. P.W.S Publishing Co., 1996, pp. 144–191.

[55] Gomory, R. "Outline of an Algorithm for Integer Solutions to Linear Programs". *Bulletin Amer. Math. Soc. 64-5* (1958), 275–278.

[56] Gomory, R., e Hu, T. "Multi-Terminal Network Flows". *SIAM Journal of Applied Mathematics 9* (1961), 551–570.

[57] Grötschel, M., Junger, J., e Reinelt, G. "A cutting plane algorithm for the linear ordering problem". *Operations Research 32* (1984), 1195–1220.

[58] Grötschel, M., e Padberg, M. "On the Symmetric Traveling Salesman Problem I: Inequalities". *Mathematical Programming 16* (1979), 265–280.

[59] Grötschel, M., e Padberg, M. "Polyhedral Theory". In *The Traveling Salesman Problem*, E. Lawler, J. Lenstra, A. R. Kan, e D. B. Shmoys, Eds. John Wiley and Sons Ltd., 1985, pp. 251–305.

[60] Grötschel, M., e Pulleyblank, W. "Clique Tree Inequalities and the Symmetric Traveling Salesman Problem". *Mathematics of Operations Research 11* (1986), 537–569.

[61] Guignard, M. "Efficient cuts in Lagrangian Relax-and-Cut schemes". *European Journal of Operational Research 105* (1998), 216–223.

[62] Guignard, M. "Lagrangian Relaxation". *TOP - Sociedad de Estadística e Investigación Operativa 11(2)* (2003), 151–228.

[63] Gusfield, D. "Very simple methods for all pairs network flow analysis". *SIAM Journal on Computing 19(1)* (1990), 143–155.

[64] Hall, L., e Magnanti, T. "A polyhedral intersection theorem for capacitated spanning trees". *Mathematics of Operations Research 17(2)* (1982), 398–410.

[65] Hao, J., e Orlin, J. B. "A Faster Algorithm for Finding the Minimum Cut in a Directed Graph". *J. Algorithms 17* (1994), 424–446.

[66] Haouari, M., e Siala, J. C. "A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem". *Computers and Operations Research 33* (2006), 1274–1288.

[67] Held, M., e Karp, R. "The Traveling Salesman Problem and Minimum Spanning Trees". *Operations Research 18* (1970), 1138–1162.

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

[69] Held, M., Wolfe, P., e Crowder, H. "Validation of Subgradient Optimization". *Mathematical Programming 6* (1974), 62–88.

[70] Johnson, D., Minkoff, M., e Phillips, S. "The prize collecting Steiner tree problem: Theory and Practice". In *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms* (2000).

[71] Karp, R. "Reductibility among combinatorial problems". In *Complexity of Computer Computations*, E. Miller e J. Thatcher, Eds. Plenum Press, 1972, pp. 85–103.

[72] Karp, R. M., e Papadimitriou, C. H. "On linear characterizations of combinatorial optimization problems". *SIAM Journal on Computing 11* (1982), 620–632.

[73] Khuller, S., Raghavachari, B., e Young, N. "Low degree spanning trees of small weight". *SIAM Journal on Computing 25 (2)* (1996), 355–368.

[74] Knowles, J., e Corne, D. "A new evolutionary approach to the degree constrained minimum spanning tree problem". *IEEE Transactions on evolutionary computation 4(2)* (2000), 125–134.

[75] Koch, T., e Martin, A. "Solving Steiner Tree Problems in Graphs to Optimality". *Networks 32* (1998), 207–232.

[76] Krishnamoorthy, M., Ernst, T., e Sharaiha, Y. "Comparison of Algorithms for the Degree constrained minimum spanning tree". *Journal of Heuristics 7* (2001), 587–611.

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

[78] Land, A., e Doig, A. "An Automatic Method for Solving Discrete Programming Problems". *Econometrica 28* (1960), 497–520.

[79] Legües, A., Ferland, J., Ribeiro, C., Vera, J., e Weintraub, A. "A Tabu Search approach for solving a difficult forest harvesting machine location problem". *European Journal of Operational Research* (2006). In press.

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

[81] Lemarechal, C. "Lagrangian Relaxation". In *Computational Combinatorial Optimization*, M. Jünger e D. Naddef, Eds. Springer, Berlin, 2001, pp. 112–156.

[82] Letchford, A., Reinelt, G., e Theis, D. "A faster exact separation algorithm for blossom inequalities". In *Integer Programming and Combinatorial Optimization 10. Lecture Notes in Computer Science*, G. Nemhauser e D. Bienstock, Eds., vol. 3064. Springer Verlag, Berlin, 2004.

[83] Letchford, A. N., e Lodi, A. "Polynomial-Time Separation of Simple Comb Inequalities". In *IPCO 2002*, W. Cook e A. S. Schulz, Eds., vol. LNCS 2337. Springer-Verlag, 2002, pp. 93–108.

[84] Little, J. D. C., Murty, K. G., Sweeney, D., e Karel, C. "An algorithm for the traveling salesman problem". *Operations Research 11* (1963), 972–989.

[85] Ljubic, I., Weiskircher, R., Pferschy, U., Klau, G., Mutzel, P., e Fischetti, M. Solving the Prize-Collecting Steiner Problem to Optimality. Relatório técnico, Technische Universität Wien, Institut fur Computergraphik und Algorithmen, 2004.

[86] Lucena, A. Tight bounds for the Steiner problem in graphs. Talk given at the TIMS-XXX - SOBRAPO XXIII Joint International Meeting, Rio de Janeiro, July 15-17 1991.

[87] Lucena, A. "Steiner Problem in Graphs: Lagrangean Relaxation and Cutting Planes". *COAL Bulletin 21* (1992), 2–8.

[88] Lucena, A. Tight bounds for the Steiner problem in graphs. Relatório Técnico TR-21/93., Dipartimento di Informatica. Univesitá degli Studi di Pisa, Pisa, Italy, 1993.

[89] Lucena, A. "Non Delayed Relax-and-Cut Algorithms". *Annals of Operations Research 140* (2005), 375–410.

[90] Lucena, A., e Beasley, J. "A Branch and Cut Algorithm for the Steiner Problem in Graphs". *Networks 31* (1998), 39–59.

[91] Lucena, A., e Beasley, J. "Branch-and-Cut algorithms". In *Advances in Linear and Integer Programming*, J. Beasley, Ed. Oxford Univeristy Press, 1998, pp. 187–221.

[92] Lucena, A., e Resende, M. "Strong lower bounds for the prize-collecting Steiner problem in Graphs". *Discrete Applied Mathematics 141* (2004), 277–294.

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

[94] Magnanti, T., e Wolsey, L. "Optimal Trees". In *Handbooks in OR and MS*, O. B. et al, Ed., vol. 7. North-Holland, Amsterdam, 1995, pp. 503–615.

[95] Margot, F., Prodon, A., e Liebling, T. M. "Tree polytope on 2-trees". *Mathematical Programming 63* (1994), 183–192.

[96] Martinhon, C., Lucena, A., e Maculan, N. "Stronger K-tree relaxations for the vehicle routing problem". *European Journal of Operational Research 158* (2004), 56–71.

[97] Miliotis, P. "Using cutting planes to solve the Symmetric Traveling Salesman Problem". *Mathematical Programming 15* (1978), 177–188.

[98] Minoux, M. *Mathematical Programming: Theory and Algorithms.* John Wiley and Sons, 1986.

[99] M.Minkoff. The Prize Collecting Steiner Tree Problem. Tese de Mestrado, Departament of Electrical Engineering and Computer Science, Massachusetts Insitute of Technology, 2000.

[100] NADDEF, D., E THIENEL, S. "Efficient separation routines for the symmetric traveling salesman problem I: general tools and comb separation". *Mathematical Programming 92* (2002), 237–255.

[101] NADDEF, D., E THIENEL, S. "Efficient separation routines for the symmetric traveling salesman problem II: separating multi handle inequalities". *Mathematical Programming 92* (2002), 257–283.

[102] NARULA, S., E HO, C. "Degree-constrained minimum spanning tree". *Computers and Operations Research 7* (1980), 239–249.

[103] NEMHAUSER, G., E SIGISMONDI, G. "A strong cutting plane branch-and-bound algorithm for node packing". *J. Oper Res Soc 43* (1992), 443–457.

[104] NEMHAUSER, G., E WOLSEY, L. *Integer and Combinatorial Optimization.* Wiley Interscience, 1988.

[105] PADBERG, M., E GRÖTSCHEL, M. "Polyhedral Computations". In *The Traveling Salesman Problem*, E. Lawler, J. Lenstra, A. R. Kan, e D. B. Shmoys, Eds. John Wiley and Sons Ltd., 1985, pp. 251–305.

[106] PADBERG, M., E HONG, S. "On the symmetric traveling salesman problem: A computational study". *Mathematical Programming Study 12* (1980), 78–107.

[107] PADBERG, M., E RAO, M. "Odd minimum cut-sets and b-matchings". *Mathematics of Operations Research 7* (1982), 67–80.

[108] PADBERG, M., E RINALDI, G. "A Branch-and-Cut algorithm for resolution of large scale of Symmetric Traveling Salesman Problem". *SIAM Review 33* (1991), 60–100.

[109] PADBERG, M., E WOLSEY, L. "Trees and cuts". *Annals of Discrete Mathematics 17* (1983), 511–517.

[110] PAPADIMITRIOU, C., E VAZIRANI, U. "On two geometric problems related to the Traveling Salesman Problem". *Journal of Algorithms 5* (1984), 231–246.

[111] PRIM, R. "Shortest connection networks and some generalizations". *Bell System Tech. J. 36* (1957), 1389–1401.

[112] Queyranne, M., e Wang, Y. "Hamiltonian path and symmetric traveling salesman polytopes". *Mathematical Programming 58* (1993), 89–110.

[113] Ribeiro, C., e de Souza, M. "Tabu Search for the Steiner Problem in Graphs". *Networks 36(2)* (2000), 138–146.

[114] Ribeiro, C., e de Souza, M. "Variable Neighborhood Search for the Degree-Constrained Minimum Spanning Tree Problem". *Discrete Applied Mathematics 118* (2002), 43–54.

[115] Rosseti, I., de Aragão, M. P., Ribeiro, C., Uchoa, E., e Werneck, R. "New benchmark instances for the Steiner problem in graphs". In *Extended Abstracts of the 4th Metaheuristics International Conference* (2001), pp. 557–561.

[116] Rönnqvist, M. "Optimization in forestry". *Mathematical Programming - Series B 97* (2003), 267–284.

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

[118] Segev, A. "The Node-Weighted Steiner Tree Problem". *Networks 17* (1987), 1–17.

[119] Shapiro, J. F. "A survey of Lagrangean techniques for Discrete Optimization". *Annals of Discrete Mathematics 5* (1978), 113–138.

[120] Takahashi, H., e Matsuyama, A. "An approximate solution for the Steiner Problem in Graphs". *Math. Japonica 24 (6)* (1980), 573–577.

[121] Vera, J., Weintraub, A., Koenig, M., Bravo, G., Guignard, M., e Barahona, F. "A Lagrangian relaxation approach for a machinery location problem in forest harvesting". *Pesquisa Operacional 23(1)* (2003), 111–128.

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

[123] Weintraub, A. Comunicação pessoal.

[124] WINTER, P. "Steiner Problem in Networks: a survey". *Networks 17* (1987), 129–167.

[125] WOLSEY, L. *Integer Programming.* John Wiley and Sons, 1998.

[126] YAMAMOTO, Y. "The Held-Karp algorithm and Degree-Constrained Minimum 1-trees". *Mathematical Programming 15* (1978), 228–231.

[127] ZHOU, G., E GEN, M. "A note on genetic algorithms for degree constrained spanning tree problems". *Networks 30* (1997), 91–95.

Apêndice A

Separação exata de desigualdades de eliminação de subrotas

Neste Apêndice, apresentamos um algoritmo introduzido por Padberg e Wolsey [109] para a separação exata das desigualdades de eliminação de subrotas SECs. Na verdade, apresentaremos apenas o principal resultado teórico que fundamenta o algoritmo. Este, propriamente, ficará implícito a partir desse resultado. Na forma como apresentamos aqui, o resultado pode ser utilizado para a separação exata de uma classe de desigualdades mais geral, as desigualdades de eliminação de subrotas generalizadas GSECs (veja [92]). Para tanto, considere uma formulação expandida para o problema de encontrar uma árvore de mínimo custo, não necessariamente geradora, para o grafo $G = (V, E)$:

$$\min \left\{ \sum_{e \in E} c_e x_e : (x, y) \in ST_e \cap \mathbb{B}^{|E|+|V|} \right\}, \tag{A.1}$$

onde o poliedro $ST_e \in \mathbb{R}^{|V|+|E|}$ é dado por:

$$x(E(S)) \leq y(S \setminus \{l\}),\ \forall l \in S,\ \forall S \subset V,\ S \neq \emptyset, \tag{A.2}$$

$$0 \leq x_e \leq 1,\ \forall e \in E, \tag{A.3}$$

$$0 \leq y_i \leq 1,\ \forall i \in V. \tag{A.4}$$

Na formulação acima, x denota o vetor de incidência das arestas escolhidas para fazer parte da árvore e y denota o vetor de incidência associado aos vértices que induzem arestas da mesma. Note que, como formulado acima, diante de custos não negativos para as arestas, uma solução ótima para (A.1) será um vértice isolado. Assim sendo, em um lapso de formalismo, vamos considerar vértices isolados como possíveis árvores de G.

Alguns casos particulares de (A.2)-(A.4) merecem ser destacados. Se $y_i = 1, \forall i \in V$, isto é, se todos os vértices devem ser cobertos, (A.2)-(A.4) em conjunto com o hiperplano $x(E) = y(V) - 1$ define o politopo das árvores geradoras [36]. Se $y_i = 1$ para um sub-conjunto $S \subset V$ de vértices (por exemplo o conjunto de vértices *terminais* do Problema de Steiner em Grafos), ou se todos os vértices i são livres para induzirem arestas da árvore (como no caso do Problema de Steiner em Grafos com Recolha de Prêmios), (A.2)-(A.4) não define os politopos correspondentes.

A garantia de separação polinomial das GSECs (A.2) baseia-se no fato de que, se resolvermos, para um ponto $(\overline{x}, \overline{y})$ satisfazendo (A.3)-(A.4), o problema

$$\min_{\emptyset \neq S \subset V} \{\overline{y}(S \setminus \{l\}) - \overline{x}(E(S))\}, \tag{A.5}$$

estaremos também resolvendo o problema de atestar se existe ou não uma desigualdade (A.2) violada por $(\overline{x}, \overline{y})$.

Vamos mostrar primeiro que resolver (A.5) equivale a resolver alguns problemas de fluxo máximo. Para este propóstio, vamos definir uma rede capacitada $N = (V \cup \{s, t\}, A)$, construída da seguinte forma:

1. para cada aresta $(i, j) \in E$ tal que $0 < \overline{x}_{(i,j)} \leq 1$, criamos dois arcos $[i, j]$ e $[j, i]$ em A, com capacidade igual a $\frac{1}{2}\overline{x}_{(i,j)}$;

2. introduzimos em N um vértice origem s e arcos $\{[s, i] : i \in V\}$, com capacidade igual a $\max\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\}$;

3. introduzimos em N um vértice destino t e arcos $\{[i, t] : i \in V\}$, com capacidade igual a $\max\{\overline{y}_i - \frac{1}{2}\overline{x}(\delta(i)), 0\}$.

Seja então $S \cup \{s\}$ um corte mínimo em N e $C(S \cup \{s\}, (V \cup \{t\}) \setminus S)$ sua capacidade. Note que na descrição a seguir, $\overline{x}(S, V \setminus S)$ é utilizado para denotar a soma

$\sum_{(i,j)\in E, i\in S, j\in V\setminus S} \overline{x}_{(i,j)}$. Veja então que, para $S \neq \emptyset$:

$$\begin{aligned}
C(S\cup\{s\},(V\cup\{t\})\setminus S) &= \sum_{i\in V\setminus S} \max\left\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\right\} + \\
&\quad \sum_{i\in S} \max\left\{\overline{y}_i - \frac{1}{2}\overline{x}(\delta(i)), 0\right\} + \\
&\quad C(S, V\setminus S) = \\
&= \sum_{i\in S} \max\left\{\overline{y}_i - \frac{1}{2}\overline{x}(\delta(i)), 0\right\} - \\
&\quad \sum_{i\in S} \max\left\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\right\} + \\
&\quad \tfrac{1}{2}\overline{x}(S, V\setminus S) + \sum_{i\in V} \max\left\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\right\} = \\
&\quad \overline{y}(S) - \tfrac{1}{2}\sum_{i\in S}\overline{x}(\delta(i)) + \\
&\quad \tfrac{1}{2}\overline{x}(S, V\setminus S) + \sum_{i\in V} \max\left\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\right\} = \\
&= \overline{y}(S) - \overline{x}(E(S)) + \sum_{i\in V} \max\left\{\frac{1}{2}\overline{x}(\delta(i)) - \overline{y}_i, 0\right\}.
\end{aligned} \tag{A.6}$$

Note que o último termo da expressão acima independe de S. Para garantir a condição $S \neq \emptyset$, basta calcularmos $|V|$ cortes mínimos, forçando, em cada um deles, a presença de um vértice de V distinto no corte. Para que um vértice, digamos l, necessariamente pertença ao corte mínimo desejado, basta alterarmos a capacidade do arco $[s, l]$ para ∞. Em cada problema de fluxo máximo (corte mínimo) a ser resolvido, as capacidades dos demais arcos permanecem como definidas anteriormente. Desta forma, mostramos que resolver (A.5) equivale, de fato, a resolver uma série de problemas de fluxo máximo (corte mínimo) em uma rede convenientemente escolhida. Dentre os $|V|$ cortes gerados aquele com a menor capacidade corresponderá a uma solução ótima de (A.5).

Apêndice B

A separação exata de desigualdades Blossom

A separação exata de desigualdades Blossom foi originalmente estudada por Padberg e Rao [107] que sugeriram um algoritmo polinomial de complexidade $O(|V|^5)$ para resolver o problema. Recentemente, um algoritmo mais eficiente (aqui denominado de LRT), de complexidade $O(|V|^4)$, foi introduzido por Letchford et al. [82]. Uma vez que LRT é empregado no Capítulo 4 para determinar os limites z_{BLO}, esse algoritmo será aqui descrito. Assumimos que as desigualdades (3.4), presentes em número polinomial na descrição poliédrica de $CM(n, b, u)$, sejam satisfeitas por $\overline{x}$, o vetor a ser separado. Diante disto, separar $\overline{x}$, isto é, responder se $\overline{x} \in CM(n, b, u)$, equivale a dizer se existe uma desigualdade Blossom violada por $\overline{x}$.

Antes de apresentar formalmente o algoritmo de separação, é necessário introduzir o conceito do que seja uma *cut-tree* de um grafo capacitado $G = (V, E)$. O conceito de cut-tree foi introduzido em Gomory e Ho [56] ao mostrar que os valores de fluxo máximo entre os $\frac{|V|(|V|-1)}{2}$ pares de vértices de V podem ser obtidos através da solução de apenas $|V| - 1$ problemas de fluxo em redes. Os $|V| - 1$ valores associados às soluções desses problemas, bem como os cortes mínimos por eles induzidos, estão representados de uma forma bastante compacta em uma estrutura denominada cut-tree, formalmente definida a seguir.

Definição B.1 *Considere um grafo $G = (V, E)$ no qual capacidades são associadas às arestas de E. Uma cut-tree de G é uma árvore $\mathcal{T}$ que conecta todos os vértices de G. As arestas de $\mathcal{T}$, no entanto, podem ou não existir no grafo G e têm capacidades associadas, não necessariamente iguais às capacidades das arestas de G que lhes sejam eventualmente correspondentes. $\mathcal{T}$ deve satisfazer às seguintes propriedades:*

- *o fluxo máximo em G entre os vértices $i, j \in V$ é dado pela capacidade da aresta de menor capacidade no caminho único em $\mathcal{T}$ que leva de i a j.*

- *os vértices pertencentes às duas componentes conexas resultantes da remoção em $\mathcal{T}$ da aresta e definida acima formam, em G, um corte mínimo entre i e j.*

Note que cada aresta $e = (p, q)$ da cut-tree *representa* um problema de fluxo máximo em G. Ao eliminarmos e de $\mathcal{T}$, obtemos dois conjuntos de vértices disjuntos de V, respectivamente, $V_{\mathcal{T}}^p$ e $V_{\mathcal{T}}^q$. Estes conjuntos induzem cortes mínimos em G.

Padberg e Rao [107] mostraram que separar desigualdades Blossom equivale ao problema de determinar uma cut-tree em uma rede convenientemente escolhida. Se n e m denotam, respectivamente, o número de vértices e arestas do problema de b-matching u-capacitado, esta rede poderia ter $n+m$ vértices e envolver a resolução de $n+m$ problemas de fluxo.

Letchford et al. [82] conseguiram mostrar, no entanto, que a resolução do problema de separação de desigualdades Blossom requer apenas a construção de uma cut-tree associada a uma rede cujo número de vértices é da mesma ordem de grandeza de n. Desta forma, obtiveram um algoritmo de separação de complexidade inferior à do algoritmo proposto em [107]. Antes de descrevermos LRT, considere a rede $G' = (V', E')$, com pesos w associados às suas arestas, definida como:

- $V' = V \cup \{r\}$, onde r é um *vértice artificial*;

- $E' = \{e \in E : \overline{x}_e \in (0, u_e]\} \cup \{(r, i) : i \in V\}$;

- e, finalmente,

$$w_{(i,j)} = \begin{cases} \min\{\overline{x}_{(i,j)}, u_{(i,j)} - \overline{x}_{(i,j)}\}, \text{ se } \overline{x}_{(i,j)} \in (0, u_e] \text{ para } (i,j) \in E \\ b_i - \overline{x}(\delta(i)), \text{ se } (i,j) \in E' \setminus E. \end{cases} \quad \text{(B.1)}$$

Observe que, na rede acima, os vértices oriundos de V se conectam ao vértice artificial r através de uma aresta cuja capacidade equivale à folga das desigualdades (3.4). As capacidades das demais arestas de E' assumem valores iguais a $\overline{x}_e$ ou iguais a $(\overline{x} - u_e)$. Se uma aresta $e \in E$ é tal que $w_e = \overline{x}_e$, dizemos que esta aresta é *normal*. Caso contrário, a aresta é denominada *complementar*.

O principal resultado teórico obtido em [82] é: $\overline{x} \notin CM(n, b, u)$ se e somente se existe uma desigualdade Blossom violada, induzida por algum handle definido

pelos conjuntos de vértices obtidos ao se eliminar uma das $|V'|-1$ arestas de uma cut-tree associada a G'. Em outras palavras, se desejamos separar desigualdades Blossom, todos os conjuntos de handles cuja investigação precisa ser considerada são definidos pelos cortes mínimos representados pelas arestas de uma cut-tree de G'. Para responder se $\overline{x} \notin CM(n,b,u)$, é necessário, portanto, resolver n problemas de fluxo máximo na rede G', que possui $n+1$ vértices e, no máximo, $m+n$ arestas.

Para melhor compreender o algoritmo de [82], considere a Definição a seguir.

Definição B.2 *Assuma que (p,q) é uma aresta da cut-tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ de G'. Considere ainda que $V_{\mathcal{T}}^p$ é o conjunto de vértices obtido ao se eliminar a aresta (p,q) de $\mathcal{T}$, tal que $r \notin V_{\mathcal{T}}^p$. No contexto do algoritmo LRT, dizemos que a desigualdade Blossom induzida por $V_{\mathcal{T}}^p$ é ímpar se*

$$\sum_{i \in V_{\mathcal{T}}^p} b_i + \sum_{\substack{e \in \delta(V_{\mathcal{T}}^p) \subseteq E: \\ w_e = u_e - \overline{x}_e}} u_e$$

é ímpar.

Note que o primeiro termo da soma acima representa a soma ($b(V_{\mathcal{T}}^p)$) dos limites de grau para os vértices em $V_{\mathcal{T}}^p$, enquanto que o segundo termo representa a soma dos u_e's associados às arestas complementares em $\delta(V_{\mathcal{T}}^p)$.

Observe que se $V_{\mathcal{T}}^p$ induz uma desigualdade ímpar e a capacidade da aresta (p,q) na cut-tree é inferior à unidade (isto é, se o fluxo máximo entre p e q em G' é inferior à unidade) uma desigualdade Blossom violada foi encontrada. Basta fazer $H = V_{\mathcal{T}}^p$ e definir T como o conjunto de arestas complementares que pertencem a $\delta(V_{\mathcal{T}}^p)$. Isto pode ser facilmente verificado ao se observar a desigualdade Blossom na forma $x(\delta(H) \backslash T) + \sum_{e \in T \subseteq \delta(H)} (u_e - x_e) + \sum_{i \in H} (b_i - x(\delta(i)) \geq 1$. Note que, para minimizar o lado esquerdo desta desigualdade, uma aresta $e \in \delta(V_{\mathcal{T}}^p)$ deverá pertencer a T ou a $\delta(V_{\mathcal{T}}^p) \backslash T$ dependendo do valor de $\min\{\overline{x}_e; u_e - \overline{x}_e\}$. Assim sendo, as arestas complementares em $\delta(V_{\mathcal{T}}^p)$ são candidatas naturais para integrar o conjunto de arestas T que define a desigualdade. Qualquer outra escolha acarreta uma desigualdade Blossom menos violada.

Caso a desigualdade não seja ímpar, indentificamos a aresta em $\delta(V_{\mathcal{T}}^p)$ para a qual u_e é ímpar, cuja adição (resp. exclusão) em (de) T (como definido acima) aumenta minimamente o lado esquerdo da desigualdade Blossom. Isto é feito da seguinte forma. Determinamos a aresta $f \in \delta(V_{\mathcal{T}}^p)$ que minimiza a diferença $|\overline{x}_e -$

$(u_e - \overline{x}_e)|, \forall e \in \delta(V_{\mathcal{T}}^p) \subset E$, tal que $u_e \bmod 2 = 1$. Se $f \in T$, fazemos $T \leftarrow T \setminus \{f\}$, caso contrário, $T \leftarrow T \cup \{f\}$. Obviamente, após a redefinição de T, a desigualdade Blossom induzida por H e T é ímpar.

Este procedimento de encontrar o conjunto ótimo de arestas em $\delta(V_{\mathcal{T}}^p)$ para definir T, de forma que a desigualdade Blossom resultante seja ímpar, foi denominado em [82] de **verificação** do handle $V_{\mathcal{T}}^p$. Tal procedimento será utilizado no Passo 5 do Algoritmo B.1, que formalmente apresenta o algoritmo LRT.

Algoritmo B.1 Algoritmo LRT para separação exata de desigualdades Blossom [82]

1: EXISTE = FALSO
2: **Construa** a rede capacitada $G' = (V', E')$
3: **Determine** uma cut-tree $\mathcal{T} = (V_{\mathcal{T}}, E_{\mathcal{T}})$ de G'
4: **para todo** $e = (p, q) \in E_{\mathcal{T}}$ **faça**
5: **Verifique** o conjunto $V_{\mathcal{T}}^p$, obtendo o conjunto T convenientemente (tal que $u(T) + b(V_{\mathcal{T}}^p)$ é ímpar).
6: **Se** o handle $V_{\mathcal{T}}^p$ e o tooth T induzem uma desigualdade violada **então**
7: EXISTE = VERDADEIRO
8: **fim Se**
9: **fim para**
10: **Retorne**(EXISTE)

Cabe ainda ressaltar que a complexidade do Algoritmo B.1 é dominada pelo cálculo de uma cut-tree de G' ($O(|V|^4)$), já que o passo 2 e os passos 4-8 do procedimento B.1 podem ser executados em tempos proporcionais a $O(m)$ e $O(nm)$, respectivamente. Para calcular a cut-tree $\mathcal{T}$ de G', empregamos o algoritmo introduzido por Gusfield [63].

Finalmente, salientamos que o Algoritmo B.1 pode ser aplicado a uma rede G' mais esparsa que a definida previamente. Isto se aplica já que as arestas $\{e \in E : \overline{x}_e = u_e\}$ não necessitam definir arestas da rede G'. Note que se uma destas arestas pertence a $\delta(V_{\mathcal{T}}^p)$, ela deve necessariamente pertencer a T para que uma desigualdade violada exista. Assim sendo, a rede G' pode ser construída através da introdução do vertice artificial r, das arestas artificiais $\{(r, i) : i \in V\}$ e de $\{e \in E : \overline{x}_e \in (0, u_e)\}$, com pesos definidos como anteriormente. Eventualmente, a rede G' resultante pode ser desconexa. Para acelerar o procedimento de separação exata, podemos aplicar o Algoritmo B.1 a cada uma das componentes conexas da rede G', em separado.

Nomenclatura e notação

- AGM: Árvore geradora de custo mínimo
- AM: Algoritmo de Minkoff
- ASRP: Árvore de Steiner com Recolha de Prêmios
- BL: Busca Local
- CRPL: Custos reduzidos de Programação Linear
- DRC: algoritmo *Delayed-Relax-and-Cut*
- GSEC: desigualdade de eliminação de subrotas generalizada
- MS: Método do Subgradiente
- NDRC: algoritmo *Non-Delated-Relax-and-Cut*
- PAGM: Problema da Árvore Geradora de Custo Mínimo
- PARG: Problema da Árvore Geradora de Custo Mínimo com Restrição de Grau nos Vértices
- PASRP: Problema da Árvore de Steiner com Recolha de Prêmios
- PCV: Problema do Caixeiro Viajante
- PDL: Problema Dual Lagrangeano
- PI: Problema de Otimização Linear Inteiro, na forma de minimização.
- PL: Programação Linear
- PL(λ): Problema Lagrangeano
- PPEM: Problema de Planejamento da Extração de Madeira

- PSG: Problema de Steiner em Grafos
- SEC: desigualdade de eliminação de subrotas
- SMMAX: Número máximo de iterações no MS
- G: grafo não direcionado
- V: conjunto de vértices de G
- E: conjunto de arestas de G
- n: cardinalidade de V
- m: cardinalidade de E
- λ: vetor de multiplicadores de Lagrange
- z: valor da função objetivo ótima de um PI
- z_λ: Limite inferior para z, dado o vetor de multiplicadores λ
- z_d: valor ótimo de PDL
- $\overline{z}$: Limite superior para z
- ρ: Parâmetro de controle da dimensão do passo no MS
- φ: Parâmetro de sobreposição no MS
- ξ: Freqüência de redução no MS