MODELOS E ALGORITMOS DE PROGRAMAÇÃO INTEIRA NO PROJETO DE REDES DE TELECOMUNICAÇÕES

Elder Magalhães Macambira

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. Nelson Maculan Filho, D.Sc.

Prof. Geraldo Robson Mateus, D.Sc.

Prof. Marcus Vinicius Soledade Poggi de Aragão, Ph.D.

Prof. Luiz Satoru Ochi, D.Sc.

Prof. Adilson Elias Xavier, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
MAIO DE 2003

MACAMBIRA, ELDER MAGALHÃES

Modelos e algoritmos de programação inteira no projeto de redes de telecomunicações [Rio de Janeiro] 2003

XIV, 184 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2003)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1 - Programação inteira

2 - Geração de colunas

3 - Planos-de-cortes

4 - Projeto de redes de telecomunicações

I. COPPE/UFRJ II. Título (série)

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

MODELOS E ALGORITMOS DE PROGRAMAÇÃO INTEIRA NO PROJETO DE REDES DE TELECOMUNICAÇÕES

Elder Magalhães Macambira

Maio/2003

Orientadores: Nelson Maculan Filho

Cid Carvalho de Souza

Programa: Engenharia de Sistemas e Computação

Esta tese dá enfâse ao estudo dos problemas que surgem durante o projeto de redes de telecomunicações. Mais precisamente, estudaremos problemas que surgem no projeto de uma rede *backbone*.

O primeiro problema, denominado Problema de Atribuição de Localidades a Anéis, caracteriza-se por ser um problema de particionamento de um grafo. São apresentadas e discutidas diferentes formulações de programação linear inteira para o problema. Investigamos a estrutura facial do poliedro associado ao problema e introduzimos novas famílias de facetas. Realizamos, ainda, um estudo sobre o problema de simetria entre as soluções do problema de atribuição de localidade a anéis. Por último, descrevemos os experimentos computacionais realizados com um algoritmo *branch-and-price* que propusemos neste trabalho. Através deste algoritmo foi possível resolver, de forma exata, instâncias com até 50 localidades.

O segundo problema, denominado de Problema de Construção dos Anéis Idênticos, pode ser reduzido uma instância do Problema do Caixeiro Viajante Simétrico. Apresentamos alguns planos-de-corte, denominados planos-de-corte geométricos, que podem ser aplicados na resolução deste problema. Estabelecemos, ainda, uma relação entre estes planos-de-corte e os planos-de-corte canônicos. Em seguida, relatamos alguns resultados computacionais obtidos com o emprego do algoritmo *cut-and-branch* que propusemos, para resolver instâncias do problema do caixeiro viajante simétrico. O desempenho computacional deste algoritmo mostra a força dos planos-de-corte geométricos e, assim, a possibilidade do seu emprego na resolução de instâncias do problema de construção dos anéis idênticos.

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

MODELS AND ALGORITHMS IN INTEGER PROGRAMMING FOR THE TELECOMMUNICATION NETWORK DESIGN

Elder Magalhães Macambira
May/2003

Advisors: Nelson Maculan Filho

Cid Carvalho de Souza

Department: Systems Engineering and Computer Science

This thesis focuses on the study of the problems that come up in network design, particularly, those that occur in a backbone network. The first problem, called SONET Ring Assignment Problem, can be described as a node-partitioning problem for a given graph. Different integer programming formulations are presented and discussed. The facial structure of the polyhedron associated to the problem is investigated, and new families of facets describing inequalities are introduced. Moreover, we study the symmetry that is inherent to this problem. Finally, we describe the computational experiments carried out with a branch-and-price algorithm that we have proposed. Such algorithm has enabled us to solve to optimality instances for graphs with up to 50 sites.

After the ring assignment problem, the length of the rings can be minimized in a second phase, using the Symmetric Traveling Salesman Problem solution techniques. The problem that comes up in this phase is called Construct Ring Problem. Some cutting planes that can be applied to solving this problem are presented. Such cuting planes are called geometric cuts. A relation between geometric and canonical cuts is also established. Furthermore, we present the computational results obtained with the cut-and-branch algorithm that we proposed to solve small instances of the Symmetric Traveling Salesman Problem. These results show the computational strength of the new cut introduced in this work. Also, there is empirical evidence that geometric cuts can be used in solving the Construct Ring Problem.

"À Ana Flávia, minha querida esposa, que sempre esteve presente nos momentos tristes e felizes deste trabalho, e que é para mim, a prova viva do amor entre um homem e uma mulher."

Agradecimentos

A Deus.

Aos meus queridos pais José Edilson e Francisca, pelo amor, incentivo e compreensão.

Ao professor Nelson Maculan Filho pela estimada amizade, por sua generosidade e conselhos, e sobretudo, pelo exemplo de humanidade a ser seguido.

Ao amigo Cid Carvalho de Souza pela amizade segura, quando me acolheu novamente em Campinas, e pela dedicação e disposição sempre presentes nos trabalhos que realizamos conjuntamente.

A todos os meus amigos do Laboratório de Otimização, próximos ou distantes, que conviveram comigo durante todo esse tempo do doutorado. Muito obrigado. De maneira bastante especial, aos amigos das Repúblicas da Farinha e da Dragão do Mar, praticamente irmãos, pelo apoio e ajuda imprescindíveis durante minha caminhada no curso.

A todos funcionários e professores do Programa de Engenharia de Sistemas de Computação. De forma especial, à funcionária Fátima que teve uma atenção e paciência invejável. Muito obrigado.

Aos professores Abílio Lucena, Adilson Elias Xavier e Geraldo Robson Mateus pelas discussões, e por terem me fornecido informações para a concepção deste trabalho.

Ao CNPq e à FAPERJ pelas bolsas de estudos concedidas durante o período de doutorado.

Sumário

1 Introdução **1**

1.1 Principais objetivos 6

1.2 Organização da tese 7

1.3 Conceitos básicos 10

2 Problema de atribuição de localidades a anéis **11**

2.1 Descrição do problema 12

2.1.1 Definição e complexidade 13

2.2 Formulações inteiras 14

2.2.1 Formulação de particionamento 14

2.2.2 Formulação de empacotamento 19

2.3 Resultados computacionais 21

2.3.1 Classe C1 22

2.3.2 Classe C2 25

2.4 Considerações 30

3 Estudo poliédrico do problema de atribuição de localidades a anéis **31**

3.1 Resultados para o poliedro do SRAP 31

3.1.1 Resultados básicos sobre o politopo Q 33

3.2 Outras desigualdades válidas para Q 43

3.3 Resultados computacionais 50

3.3.1 Estratégias 51

3.4 Reduzindo a simetria das soluções 53

3.4.1 Desigualdades lineares 54

3.4.2 Resultados computacionais 56

3.5 Considerações 60

4 Limitantes superiores **61**

4.1 Trabalhos anteriores 61

4.1.1 Heurísticas gulosas 62

4.1.2 Metaheurísticas 63

4.2 Metaheurística *GRASP* 64
4.2.1 Fase de construção 67
4.2.2 Fase de busca local 68
4.3 Resultados computacionais 70
4.3.1 Classe C1 71
4.3.2 Classe C2 72
4.4 Considerações 74

5 Algoritmo *branch-and-price* para o problema de atribuição de localidades a anéis 76
5.1 Introdução 77
5.2 Reformulando o SRAP 79
5.3 Um modelo de geração de colunas 85
5.3.1 Definindo o problema mestre 85
5.3.2 Definindo o problema auxiliar 86
5.4 Estudo poliédrico do problema auxiliar 88
5.5 Visão geral do algoritmo *branch-and-price* 94
5.5.1 Conjunto inicial de colunas 95
5.5.2 Estrutura do procedimento *branch-and-bound* 95
5.5.3 Limitantes superiores 99
5.5.4 Heurísticas primais 99
5.5.5 Detecção de *tailing off* 100
5.5.6 Estratégias de seleção de colunas 100
5.6 Resultados computacionais 109
5.6.1 Classe C1 110
5.6.2 Classe C2 117
5.7 Considerações 126

6 Planos-de-corte geométricos para programação inteira 0-1 128
6.1 Preliminares 129
6.2 Planos-de-corte esféricos 131
6.2.1 Planos-de-corte 1-esférico 132
6.2.2 Planos-de-corte p-esférico 135
6.3 Planos-de-corte cilíndricos 136
6.3.1 Planos-de-corte 1-cilíndrico 137
6.3.2 Planos-de-corte p-cilíndrico 139
6.4 Forma geral dos planos-de-corte geométricos 140
6.5 Planos-de-corte canônicos 141
6.6 Desigualdades de eliminação de subciclos e planos-de-corte geométricos 143

7 Um algoritmo baseado em planos-de-corte geométricos para o PCVS **146**

7.1 Estrutura e aspectos de um algoritmo *cut-and-branch* 147

7.2 Emprego dos planos-de-corte p-esféricos 149

7.3 Heurística de Lin-Kernighan e planos-de-corte p-esféricos . . . 150

7.4 Visão geral do algoritmo *cut-and-branch* 152

7.4.1 Modelo matemático inicial 152

7.4.2 Rotinas de separação 153

7.4.3 Procedimento *branch-and-bound* 154

7.5 Resultados computacionais 155

7.5.1 Descrição das instâncias 155

7.5.2 Experimentos 155

7.6 Considerações 158

8 Conclusões e trabalhos futuros **160**

8.1 Contribuições 160

8.2 Direções futuras 164

Lista de Figuras

1.1 Uma rede de telecomunicações com dois níveis. 2
1.2 Estado inicial da rede *backbone*. 4
1.3 Projeto físico da rede *backbone*. 5
1.4 Projeto lógico da rede *backbone*. 5

2.1 Topologia em anel *SONET* com ADM e DCS. 13
2.2 Instância para o SRAP com n=25, m=49 e B=155Mbs. 15
2.3 Solução ótima para o SRAP com n=25, m=49 e B=155Mbs. . 15

3.1 Solução S1 obtida para uma instância com $n = 15$ e $B = 155Mbs$. 55
3.2 Solução simétrica à solução S1. 55

6.1 Plano-de-corte 1-esférico $x_1 + x_2 + x_3 \leq 2$ definido pelo ponto $\overline{x} = (1, 1, 1)$. 133
6.2 Plano-de-corte 1-esférico $x_1 + x_2 + x_3 \leq 2$ e $x_1 + x_2 - x_3 \leq 1$ definidos, respectivamente, pelos pontos $\overline{x} = (1, 1, 1)$ e $\overline{x}^2 = (1, 1, 0)$. 137
6.3 Plano-de-corte 1-cilíndrico $x_1 + x_2 \leq 1$ definido pelos pontos $\overline{x}^1 = (1, 1, 0)$ e $\overline{x}^2 = (1, 1, 1)$. 138

7.1 (a) Grafo $K_n = (V_n, E_n)$. (b) Solução obtida com relaxação 2-*matching* do PCVS para o grafo K_n. 149
7.2 Um exemplo de aplicação do plano-de-corte 4-esférico $x_{12} + x_{13} - x_{14} - x_{15} - x_{16} + x_{23} - x_{24} - x_{25} - x_{26} - x_{34} - x_{35} - x_{36} + x_{45} + x_{46} + x_{56} \leq 2$ na solução $\overline{x}$. 152

Lista de Tabelas

2.1 Distribuição das instâncias testadas na classe C1 com as formulações (P1) e (P2). 22
2.2 Resultados obtidos com as formulações para as instâncias geométricas da classe C1 com $B = 155Mbs$. 23
2.3 Resultados obtidos com as formulações para as instâncias geométricas da classe C1 com $B = 622Mbs$. 24
2.4 Resultados obtidos com as formulações para as instâncias aleatórias da classe C1 com $B = 155Mbs$. 26
2.5 Resultados obtidos com as formulações para as instâncias aleatórias da classe C1 com $B = 622Mbs$. 26
2.6 Distribuição das instâncias testadas na classe C2 com as formulações (P1) e (P2). 27
2.7 Resultados obtidos com as formulações para as instâncias geométricas da classe C2 com $B = 155Mbs$. 28
2.8 Resultados obtidos com as formulações para as instâncias geométricas da classe C2 com $B = 622Mbs$. 28
2.9 Resultados obtidos com as formulações para as instâncias aleatórias da classe C2 com $B = 155Mbs$. 29
2.10 Resultados obtidos com as formulações para as instâncias aleatórias da classe C2 com $B = 622Mbs$. 29

3.1 Desempenho do procedimento *branch-and-bound* para instâncias geométricas e aleatórias da classe C1, segundo a estratégia F1. 52
3.2 Desempenho do procedimento *branch-and-bound* para instâncias geométricas e aleatórias da classe C1, segundo a estratégia F2. 53
3.3 Desempenho do procedimento *branch-and-bound* para instâncias da classe C1, segundo a estratégia F1. 54
3.4 Redução da simetria para instâncias geométricas e aleatórias da classe C1, segundo a estratégia S1. 58

3.5 Redução da simetria para instâncias geométricas e aleatórias da classe C1, segundo a estratégia S2. 58
3.6 Resultados computacionais com a Estratégia S1 em instâncias geométricas e aleatórias da classe C1, com $n = 25$. 60

4.1 Resultados obtidos com as heurísticas gulosas para as instâncias da classe C1. 71
4.2 Resultados obtidos com a metaheurística *GRASP* para as instâncias da classe C1. 72
4.3 Soluções viáveis obtidas com a metaheurística *GRASP* para algumas instâncias da classe C1. 73
4.4 Distribuição das instâncias da classe C2 testadas com a metaheurística *GRASP*. 73
4.5 Resultados obtidos com a metaheurística *GRASP* para as instâncias da classe C2. 74
4.6 Soluções viáveis obtidas com a metaheurística *GRASP* para algumas instâncias da classe C2. 75
4.7 Distribuição das instâncias da classe C2, onde a metaheurística *GRASP* não obteve soluções viáveis. 75

5.1 Comparação entre as formulações (P2) e (P3) para as instâncias geométricas da classe C1. 84
5.2 Comparação entre as formulações (P2) e (P3) para as instâncias aleatórias da classe C1. 84
5.3 Comparação entre as formulações (P2) e (P3) para as instâncias aleatórias da classe C2. 85
5.4 Resultados computacionais obtidos com a resolução do problema auxiliar por heurísticas. 102
5.5 Resultados computacionais obtidos com a resolução do problema auxiliar por métodos exatos. 104
5.6 Resultados computacionais obtidos com a seleção de uma única coluna. 107
5.7 Resultados computacionais obtidos com a seleção de múltiplas colunas. 108
5.8 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C1 ($B = 155Mbs$). 113
5.9 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C1 ($B = 622Mbs$). 114

5.10 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C1 ($B = 155Mbs$). 115
5.11 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C1 ($B = 622Mbs$). 116
5.12 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 155Mbs$). 119
5.13 (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 155Mbs$). 120
5.14 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 622Mbs$). 121
5.15 (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 622Mbs$). 122
5.16 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 155Mbs$). 123
5.17 (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 155Mbs$). 124
5.18 Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 622Mbs$). 125
5.19 Comparação entre os desempenhos computacionais da formulação (P1) e do algoritmo *branch-and-price* na resolução de instâncias do SRAP. 127

7.1 Instâncias utilizadas nos experimentos computacionais. 155
7.2 Qualidade da solução obtida na relaxação inicial. 156
7.3 Resultados computacionais obtidos para # Cortes = 100. . . . 157
7.4 Resultados computacionais obtidos para # Cortes = 200. . . . 158
7.5 Resultados computacionais obtidos para # Cortes = 300. . . . 158

Lista de Algoritmos

1 Algoritmo informal para a metaheurística *GRASP* 65
2 Algoritmo informal para a fase de construção do *GRASP* . . . 68
3 Algoritmo informal para a fase de busca local do *GRASP* . . . 69
4 Algoritmo informal para a metaheurística *GRASP* 70
5 Algoritmo informal para gerar um conjunto inicial de colunas . 96
6 Algoritmo informal para um algoritmo *cut-and-branch* 148
7 Algoritmo informal para a rotina de separação 154

Capítulo 1

Introdução

À medida que a indústria de telecomunicações introduz novas tecnologias no mercado, a natureza e o volume de serviços oferecidos aos usuários tendem a mudar. Vários exemplos desses serviços combinam, por exemplo, voz, dados e vídeo. Esta mudança implica em novos aspectos no planejamento de uma rede de telecomunicações e, conseqüentemente, a necessidade de resolver problemas de otimização durante esta fase de planejamento.

Um problema que ocorra em uma rede de telecomunicações pode ser definido com o auxílio de um grafo, onde se deseja encontrar um subconjunto de vértices e (ou) arestas que satisfaça a determinados requisitos e que otimize uma função objetivo. Esta última pode estar relacionada tanto com o custo de instalação da rede, como com o atraso no envio de informações.

Uma rede de telecomunicações é usualmente composta de dois níveis (veja Soriano *et al.* [68]): uma rede *backbone* e uma rede de acesso local. A rede *backbone* facilita o tráfego de informações entre os usuários. Já a rede de acesso local é encarregada de concentrar essas informações e disponibilizá-las às localidades da rede *backbone*.

A Figura 1.1 mostra um exemplo de uma rede de telecomunicações com dois níveis. O primeiro nível, correspondente à rede *backbone*, é representado pelas linhas mais escuras. Já o segundo nível, indicando a rede de acesso local, é representado pelas linhas mais claras. Na rede de acesso local ressaltamos um terceiro nível, representado pelas linhas pontilhadas, que é a rede de distribuição. Esta rede é encarregada de conectar os usuários ao primeiro concentrador da rede de acesso local.

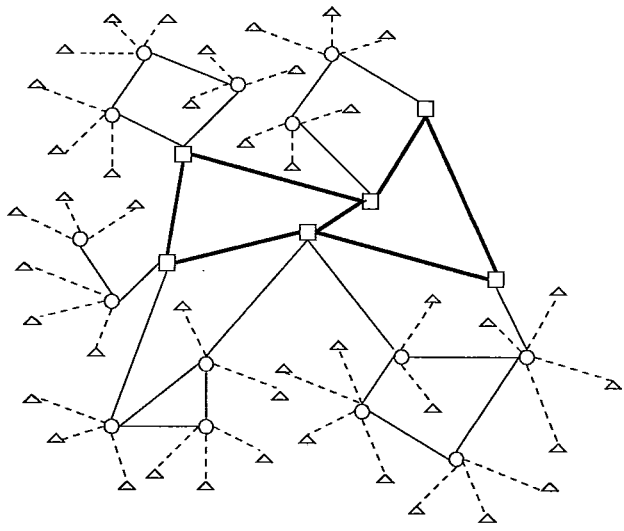

Figura 1.1: Uma rede de telecomunicações com dois níveis.

O maior interesse desta tese está relacionado ao estudo dos problemas que ocorrem no planejamento de redes de telecomunicações. Alguns destes problemas são: dimensionamento, roteamento, síntese, restauração e definição de uma topologia da rede (veja Magnanti e Wong [51]). Mais precisamente, estudaremos problemas que surgem no planejamento de uma rede *backbone*. Vejamos, em seguida, uma rápida explanação sobre esses problemas.

O planejamento de uma rede *backbone* consiste em determinar e estabelecer uma conexão entre as localidades da rede, a um custo mínimo, segundo alguns conjuntos de restrições. Um exemplo muito comum destes conjuntos de restrições nas redes atuais seriam as restrições de sobrevivência de uma rede.

A sobrevivência de uma rede corresponde a sua capacidade de permanecer ativa, no caso de ocorrer uma falha em uma localidade ou uma ligação. Esta característica de sobrevivência é explorada nas redes *backbone* através do emprego da topologia em anel com tecnologia *SONET*, do inglês *Synchronous Optical NETwork* (veja [12, 69, 76]).

No planejamento da rede *backbone*, que estudaremos neste trabalho, estamos envolvidos com a definição da topologia da rede. Em particular, nos preocupamos em adotar uma topologia em anel, segundo a tecnologia *SONET*. O planejamento dessa rede pode ser definido como se segue.

> Dado um conjunto de localidades L, pertencentes à rede de telecomunicações, uma matriz de demandas d_{uv}, com u , $v \in L$, um inteiro B representando uma capacidade de um anel, e uma função de custo c determinada por dois parâmetros: o primeiro, fixo, está associado com a atribuição de cada localidade a um anel; e o segundo, variável, está relacionado com o tamanho de cada anel, ou seja, com a menor quantidade de fibra ótica que deveremos passar para conectar as localidades de cada anel. Deseja-se determinar uma atribuição de todas as localidades a subconjuntos, que representam os anéis, tal que a capacidade desses subconjuntos seja satisfeita, de acordo com o parâmetro B, e todas as demandas entre as localidades sejam atendidas com um custo mínimo.

Nota-se que o planejamento de uma rede *backbone* é uma tarefa complexa e que usualmente pode ser dividida em etapas. Em seguida, apresentamos essas etapas:

- *projeto físico*: consiste em determinar subconjuntos de localidades que darão origem aos anéis;
- *projeto lógico*: após a escolha dos subconjuntos, o próximo passo é estabelecer uma conexão entre as localidades de cada subconjunto, ou seja, constituir os anéis a partir dos subconjuntos.

No projeto físico, a definição de uma topologia em anel de uma rede apresenta várias versões. Uma das versões mais simples dos problemas que aparecem no planejamento de uma rede *backbone*, seguindo uma topologia em anel, ocorre quando a rede possui um único anel, que atende a todas as demandas e que passa por todos os vértices (Gendreau *et al.* [25] e, Lee e Koh [39]).

Uma segunda classe corresponderia à topologia de uma rede com vários anéis. Esta classe pode ser dividida em duas subclasses: uma que aceita a interseção entre os anéis (Fortz *et al.* [23], Laguna [36], Lee *et al.* [40], Luss *et al.* [44] e Sutter *et al.* [70]), e outra que não aceita a interseção (Altinkemer [1] e Klincewicz *et al.* [35]).

Por último, podemos mencionar uma terceira classe, que envolve uma estrutura hierárquica entre os múltiplos anéis. Os anéis são considerados disjuntos. Além disso, temos a presença de um anel, denominado *anel federal*,

que realiza a conexão entre os demais anéis (Goldschmidt *et al.* [26]). Em particular, o projeto físico da rede *backbone* que estudamos neste trabalho segue a estrutura hierárquica desta classe.

A outra parte do planejamento da rede *backbone*, o projeto lógico, está associado com a quantidade de fibra ótica que deverá passar nos anéis, ligando as localidades. É evidente que essa quantidade de fibra ótica dependerá do tamanho do anel, ou seja, do menor ciclo que pudermos estabelecer entre as localidades de cada anel.

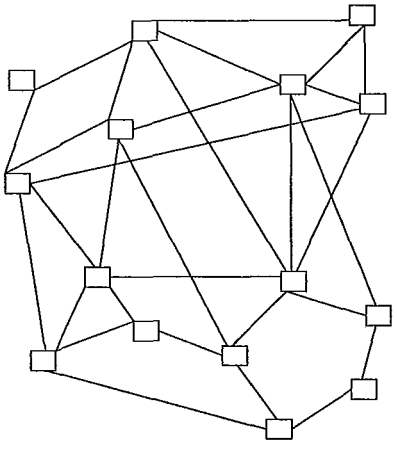

Figura 1.2: Estado inicial da rede *backbone*.

As Figuras 1.2, 1.3 e 1.4 mostram um exemplo do planejamento da rede *backbone*. A Figura 1.2 exemplifica o estado inicial da rede, ou seja, as localidades e as demandas existentes entre elas. A Figura 1.3 mostra o particionamento dessas localidades em subconjuntos. Por último, a Figura 1.4 exemplifica a conexão entre as localidades de cada subconjunto, ou seja, a formação dos anéis.

Observa-se que em cada etapa do planejamento da rede *backbone* pode-se manipular um ou mais problemas, e que os resultados obtidos em uma etapa podem ser fixados e utilizados como parâmetros de entrada para a etapa seguinte. O processo de atribuição de localidades a um anel está associado ao projeto físico. Como serão determinados agrupamentos (ou anéis) de localidades a partir da atribuição, o projeto físico pode ser visto como um problema de particionamento dos vértices de um grafo (vide Mehrotra e Trick [54]).

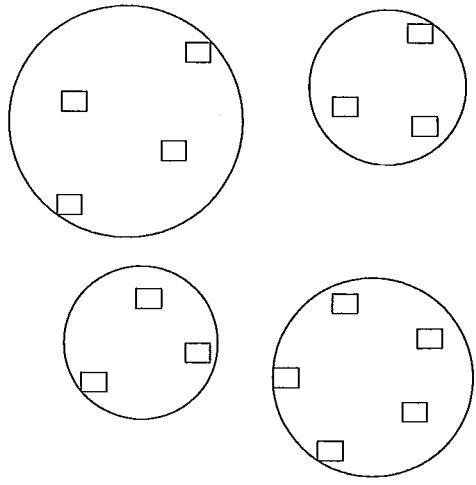

Figura 1.3: Projeto físico da rede *backbone*.

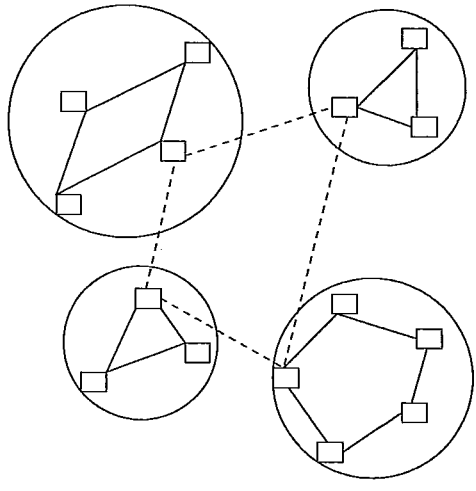

Figura 1.4: Projeto lógico da rede *backbone*.

Já na fase de construção dos anéis, ou seja, o projeto lógico, pode-se observar facilmente que tal problema corresponde a resolver uma instância do problema do caixeiro viajante simétrico (vide Jünger [33]) em cada agrupamento e, em seguida, devido à existência do anel federal, resolver uma instância do problema do caixeiro viajante simétrico generalizado (vide Fischetti *et al.* [21], e Laporte e Nobert [37]).

1.1 Principais objetivos

Observa-se que o planejamento de uma rede de telecomunicações é uma tarefa complexa, que requer a resolução de alguns problemas de otimização combinatória pertencentes à classe NP-difícil (veja Garey e Johnson [24]).

A motivação inicial para o estudo do planejamento de uma rede *backbone* veio do fato de nos determos, também, na resolução do problema envolvido com o projeto físico, diferentemente de outros trabalhos, que só tratam do projeto lógico (veja Magnanti e Wolsey [51], e Soriano *et al.* [68]). Assim, os objetivos a serem delineados serão divididos de acordo com os projetos físico e lógico.

O problema presente no projeto físico é denominado de problema de atribuição de localidades a anéis, enquanto que um dos problemas que surge no projeto lógico, relacionado com o problema do caixeiro viajante simétrico, é denominado de problema de construção dos anéis *SONET* idênticos. Estes problemas foram mencionados pela primeira vez no trabalho de Goldschmidt *et al.* [26], e percebemos que existem poucas referências que tratam da resolução destes problemas na literatura. Além do mais, estas referências dizem mais respeito ao emprego de heurísticas na resolução daqueles problemas, particularmente para o problema de atribuição de localidades a anéis.

Os principais objetivos que pretendemos alcançar com este trabalho são:

1. propor modelos matemáticos capazes de solucionar os problemas de otimização combinatória que aparecerem durante o planejamento de uma rede de telecomunicações;

2. realizar um estudo de técnicas de otimização que nos auxiliem na resolução eficiente daqueles problemas.

1.2 Organização da tese

A tese está organizada em oito capítulos. Os tópicos a serem cobertos em cada um deles são descritos a seguir.

Este primeiro capítulo foi dedicado à apresentação da estrutura básica de uma rede de telecomunicações e de alguns problemas que aparecem durante o planejamento desta rede. Além disso, nas seções anteriores, delineamos a motivação e os objetivos que desejamos alcançar com esta tese.

No segundo capítulo, trataremos do problema de atribuição de localidades a anéis. Descreveremos o problema e apresentaremos alguns modelos matemáticos para solucioná-lo. Apresentaremos, ainda, uma breve comparação entre esses modelos, através da resolução de algumas instâncias.

Na seção 2.2.2, propomos um novo modelo matemático para o problema de atribuição de localidades a anéis. Este modelo é baseado no princípio de empacotamento, em particular, o empacotamento das localidades da rede.

Na seção 2.3, apresentamos alguns experimentos computacionais na resolução do problema de atribuição de localidades a anéis. Nestes experimentos foram utilizadas uma formulação inteira existente na literatura e a formulação que estamos propondo.

Os resultados obtidos neste capítulo produziram o seguinte trabalho:

1. Problema de atribuição de localidades a anéis *SONET*: formulações inteiras 0-1 e experimentos computacionais [46].

No terceiro capítulo, apresentaremos um estudo sobre a estrutura facial do politopo associado ao problema de atribuição de localidades a anéis. Através da formulação de empacotamento foi possível definirmos novas classes de desigualdades válidas para o politopo associado a este problema. São apresentadas, também, condições para que algumas desigualdades sejam definidoras de facetas. A relevância deste estudo deve-se à inexistência de um

trabalho semelhante na literatura, isto é, que faça uma investigação poliedral do problema de atribuição de localidades a anéis.

Além disso, apresentaremos um estudo sobre a redução de simetria das soluções do problema de atribuição de localidades a anéis, através da definição de desigualdades lineares.

Na seção 3.1, realizaremos um estudo poliedral inicial do problema de atribuição de localidades a anéis. Alguns resultados básicos sobre o politopo associado a este problema, como por exemplo, a sua dimensão e a força de algumas desigualdades que estão presentes na formulação de empacotamento.

Na seção 3.2, descreveremos algumas desigualdades válidas que encontramos para o politopo associado ao problema de atribuição de localidades a anéis durante nossos experimentos computacionais. Estas desigualdades surgiram como resultado dos estudos que fizemos para obtenção de planos-de-corte que eliminassem soluções fracionárias ótimas das relaxações lineares deste politopo.

Na seção 3.4, realizaremos um estudo sobre o problema de simetria entre as soluções do problema de atribuição de localidade a anéis. Na subseção 3.4.1, nos dedicaremos a estudar como poderíamos quebrar a simetria das soluções deste problema através da inclusão de desigualdades lineares à formulação de empacotamento.

Os resultados, neste capítulo, proporcionaram a produção dos seguintes trabalhos:

1. Problema de atribuição de localidades a anéis *SONET*: formulação, desigualdades válidas e facetas [45];

2. Reduzindo a simetria no problema de atribuição de localidades a anéis *SONET* com o emprego de planos-de-corte [47].

No quarto capítulo, apresentaremos a descrição e os resultados computacionais obtidos com o emprego da metaheurística *GRASP* na obtenção de limitantes superiores para o problema de atribuição de localidades a anéis.

No quinto capítulo, descreveremos o algoritmo *branch-and-price* que estamos propondo e apresentaremos alguns resultados computacionais obtidos

com o emprego desta técnica na resolução de instâncias do problema de atribuição de localidades a anéis. A relevância do uso desta técnica na resolução, de forma exata, do problema de atribuição de localidades a anéis deve-se à inexistência de um trabalho similar na literatura.

No sexto capítulo, trataremos do problema de construção dos anéis idênticos. Apresentaremos alguns planos-de-corte, denominados planos-de-corte geométricos, que podem ser aplicados na resolução desse problema. Esses planos-de-corte são planos-de-corte de propósito geral.

Nas seções 6.2 e 6.3, apresentaremos novos planos-de-corte de caráter geral, denominados planos-de-corte esféricos e cilíndricos, para a resolução de problemas de programação inteira 0-1. São estabelecidas condições para que estes planos-de-corte definam facetas.

Na seção 6.5, estabeleceremos uma relação de *um para um* entre os planos-de-corte canônicos, definidos por Balas e Jeroslow, e os planos-de-corte geométricos que propusemos.

Na seção 6.6, mostraremos que a desigualdade de eliminação de subciclos, definida para o problema do caixeiro viajante simétrico, corresponde ao plano-de-corte geométrico que estamos propondo.

Os resultados deste capítulo produziram o seguinte trabalho:

1. Geometrical efficient cuts in integer 0-1 programming [56].

Outros trabalhos produzidos neste capítulo são os relatórios técnicos abaixo:

1. Planos-de-corte geométricos aplicados a problemas de programação linear 0-1 [55];

2. Geometrical cuts for 0-1 integer programming [57].

No sétimo capítulo, relataremos alguns resultados computacionais obtidos com o emprego do algoritmo *cut-and-branch* que propusemos para resolver instâncias de pequeno porte do problema do caixeiro viajante simétrico. A partir dos resultados obtidos com o emprego desse algoritmo, produzimos o seguinte relatório técnico:

1. Resolução do problema do caixeiro viajante simétrico com o emprego de planos-de-corte geométricos [58].

No oitavo e, último, capítulo, serão apresentadas algumas conclusões e considerações finais deste trabalho, que poderão levar a direções futuras de pesquisa.

Finalizaremos este capítulo, citando algumas referências para os conceitos básicos que empregaremos comumente no decorrer desta tese.

1.3 Conceitos básicos

Neste trabalho, partimos do princípio que o leitor tenha algum conhecimento dos conceitos básicos relacionados à otimização combinatória e programação inteira. Procuraremos seguir as definições dadas nas seguintes referências:

- Complexidade computacional: Garey e Johnson [24];
- Teoria dos grafos: Bondy e Murty [9] e, mais recentemente, o livro de West [77];
- Teoria poliedral e programação inteira: Ferreira e Wakabayashi [20], Nemhauser e Wolsey [59], e Wolsey [80].

Alguns termos técnicos em inglês, de uso consagrado e sem uma clara equivalência no português, aparecem em sua forma original no texto, como por exemplo, *branch-and-bound*, *branch-and-price* e *lifting*. Outros termos, porém, aparecem em português, como empacotamento, por exemplo, no lugar de *packing*, e particionamento, ao invés de *partitioning*.

Capítulo 2

Problema de atribuição de localidades a anéis

Neste capítulo, abordaremos um dos problemas que aparecem no planejamento de uma rede *backbone*. Esse problema consiste em particionar um conjunto de localidades da rede *backbone* dentro de anéis, em particular, anéis que seguem a tecnologia *SONET* .

Cada anel definido para a rede *backbone* deverá possuir a mesma capacidade, e cada localidade deverá ser atribuída a um único anel. O tráfego das demandas entre as localidades pertencentes a anéis diferentes ficará a cargo de um anel especial denominado anel federal. O objetivo é projetar uma rede *backbone* com um número mínimo de anéis que atendam às restrições de capacidades estipuladas para os anéis. Tal problema é denominado problema de atribuição de localidades a anéis *SONET*, ou sucintamente, problema de atribuição de localidades a anéis. Esse problema será denotado pela sigla SRAP (do inglês *SONET Ring Assignment Problem*).

Um outro tópico abordado neste capítulo é o estudo de formulações para o SRAP. Sabemos que para um dado problema é possível propormos várias formulações válidas, o que pode ser obtido simplesmente alterando-se a definição das variáveis associadas ao problema. Devido à essas diferentes maneiras de modelar um problema, torna-se interessante analisar quando uma formulação é preferível em relação a uma outra.

Assim, realizaremos um estudo de duas formulações propostas para o problema de atribuição de localidades a anéis. Esse estudo possibilitará verificar qual tipo de formulação se mostra mais adequada para este problema. A pri-

meira formulação foi proposta por Goldschmidt *et al.* [26]. Já a segunda foi proposta recentemente por nós, e baseia-se no princípio de empacotamento, ao invés de particionamento, das localidades.

2.1 Descrição do problema

Nesta seção, será apresentado o problema de atribuição de localidades a anéis que surge no planejamento de uma rede *backbone*. Este problema foi inicialmente definido em Goldschmidt *et al.* [26]. Os autores desejavam determinar um número mínimo de agrupamentos, tal que as restrições de capacidade dos agrupamentos fossem satisfeitas. Cada agrupamento de localidades obtido corresponderia a um anel *SONET* da rede *backbone*.

Para localizar o problema, vamos descrever alguns aspectos relevantes na obtenção de um número mínimo de agrupamentos em uma rede de telecomunicações com topologia em anel, segundo a tecnologia *SONET*.

Como pode ser observado na Figura 2.1, o roteamento do tráfego entre as localidades de um anel requer um tipo especial de equipamento, instalado em cada localidade, que seja capaz de agregar e desagregar a quantidade de tráfego. Este dispositivo é denominado multiplexador *Add-Drop SONET* ou, simplesmente ADM (do inglês *Add-Drop Multiplexers*).

Um ADM é classificado segundo sua capacidade:

- ADM-1: concentrador de capacidade STM-1 (155Mbs);
- ADM-4: concentrador de capacidade STM-4 (622Mbs);
- ADM-16: concentrador de capacidade STM-16 (2.5Gbs).

Todos os ADMs que compõem um anel devem possuir a mesma capacidade. Esse valor é definido como sendo a capacidade de um anel.

Adicionalmente, quando duas localidades estão localizadas em anéis diferentes, necessitamos utilizar um sistema de conexão, denominado *Digital Cross-connect System* (DCS) *SONET*, para carregar o tráfego entre as localidades.

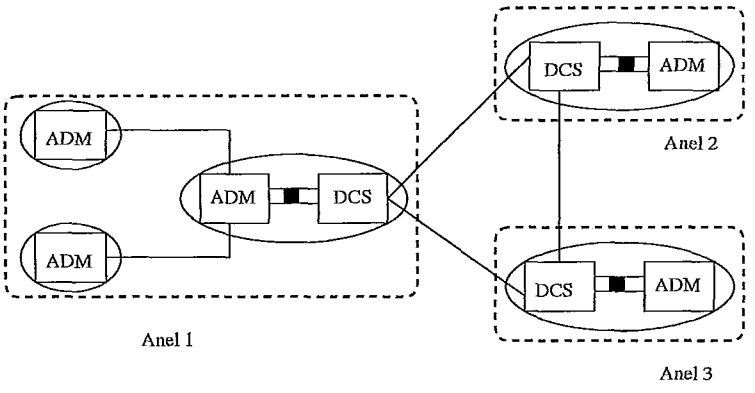

Figura 2.1: Topologia em anel *SONET* com ADM e DCS.

O principal custo envolvido com o projeto de uma rede *backbone* está diretamente relacionado ao custo dos componentes ADMs e DCSs. Já que o custo de ADMs tem diminuído significativamente nos últimos anos, a quantidade de DCSs tornou-se determinante para o custo total da rede. Desta forma, uma topologia em anel que utiliza um número mínimo de DCSs é aquela que é a mais atrativa para redes com restrições de sobrevivência.

Logo, no projeto de uma rede *backbone* pode-se considerar apenas o custo dos DCSs. Para efeito de modelagem do problema, estes custos serão considerados constantes e iguais a 1, o que é uma hipótese razoável, se assumirmos que o mesmo tipo de equipamento será instalado em todo local que necessite de um DCS.

2.1.1 Definição e complexidade

Seguindo a nomenclatura utilizada por Garey e Johnson [24], o problema de decisão definido abaixo caracteriza o problema de atribuição de localidades a anéis:

Instância: Seja $G = (V, E)$ um grafo não direcionado, onde $|V| = n$, com um inteiro d_{uv} associado a cada aresta $e = (u, v) \in E$, e B e l inteiros.

Questão: Existe uma partição do conjunto de vértices V em l subconjuntos disjuntos $V_1, \ldots, V_l$, tal que

$$\sum_{\substack{u,v \in V_i \\ u<v}} d_{uv} + \sum_{\substack{u \in V_i \\ v \notin V_i}} d_{uv} \leq B, \qquad i = 1, \ldots, l \quad \text{e,}$$

$$\sum_{i=1}^{l-1} \sum_{j=i+1}^{l} \sum_{u \in V_i} \sum_{v \in V_j} d_{uv} \leq B \ ?$$

O problema de atribuição de localidades a anéis é NP-difícil (veja Goldschmidt *et al.* [26]).

A Figura 2.2 apresenta um grafo com 25 vértices e 49 arestas. O valor estipulado para a capacidade B de cada agrupamento candidato é igual a 155Mbs. A solução para essa instância é mostrada na Figura 2.3 e composta pelos anéis 1, 4 e 10.

A partir desse ponto do texto, quando mencionarmos um vértice u do grafo G, estaremos nos referindo a uma localidade u da rede *backbone*. O mesmo ocorrerá para uma aresta (u, v), a qual indicará que existe uma demanda $d(u, v)$ entre as localidades u e v. Além disso, fica evidente que o inteiro d_{uv} associado à aresta (u, v) corresponde à demanda $d(u, v)$.

2.2 Formulações inteiras

Nesta seção, apresentamos duas formulações inteiras 0-1 para o problema de atribuição de localidades a anéis. A primeira formulação caracteriza-se por ser não linear 0−1 e se basear no princípio de particionamento das localidades. Já a segunda formulação caracteriza-se por ser linear inteira 0−1 e empregar o princípio de empacotamento das localidades. Em seguida, apresentamos e discutimos cada uma dessas formulações.

2.2.1 Formulação de particionamento

Considere o grafo não direcionado $G = (V, E)$, como definido na seção 2.1.1. Para cada localidade, definiremos a variável de decisão x_{ui} ($u \in V, i \in V$), indicando se a localidade u pertence ($x_{ui} = 1$) ou não ($x_{ui} = 0$) ao agrupamento i. Considere, ainda, a definição de uma segunda variável de decisão y_i ($i \in V$), associada a cada agrupamento i. A variável y_i indicará se o agrupamento i está ($y_i = 1$) ou não ativo ($y_i = 0$). Um agrupamento i será considerado ativo, se alguma localidade u for atribuída ao mesmo.

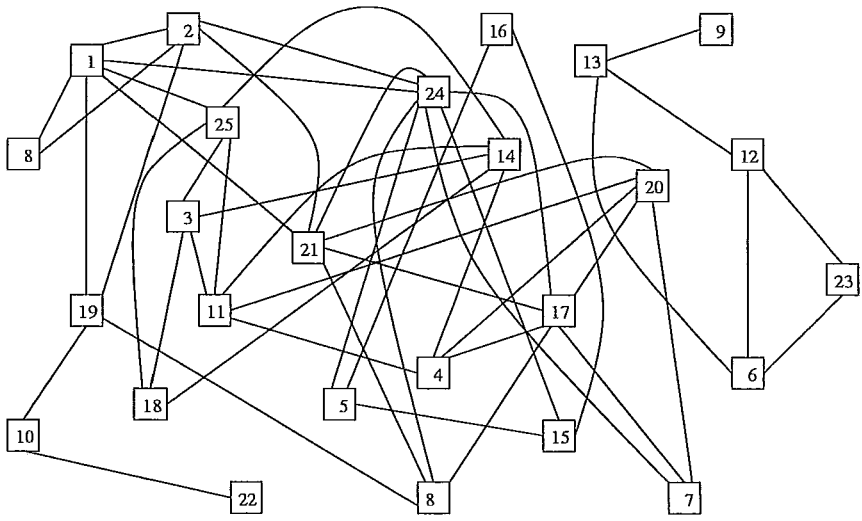

Figura 2.2: Instância para o SRAP com n=25, m=49 e B=155Mbs.

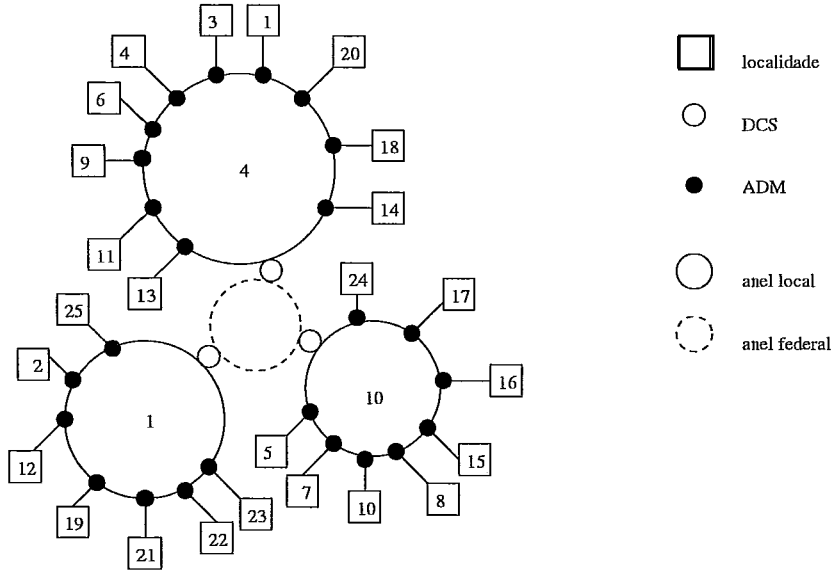

Figura 2.3: Solução ótima para o SRAP com n=25, m=49 e B=155Mbs.

Usando as variáveis de decisão definidas acima, o problema de atribuição de localidades a anéis pode ser expresso pelo seguinte problema de programação inteira 0-1:

(P1)

$$\text{minimize} \sum_{i=1}^{n} y_i, \tag{2.1}$$

sujeito a:

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv} x_{ui} x_{vi} + \sum_{u=1}^{n} \sum_{v=1}^{n} d_{uv} x_{ui} (1 - x_{vi}) \leq B, \qquad \forall i \in V, \tag{2.2}$$

$$\sum_{u=1}^{n} \sum_{v=1}^{n} \sum_{i=1}^{n-1} \sum_{j=1}^{n} d_{uv} x_{ui} x_{vj} \leq B, \tag{2.3}$$

$$\sum_{i=1}^{n} x_{ui} = 1, \qquad \forall u \in V, \tag{2.4}$$

$$x_{ui} - y_i \leq 0, \qquad \forall u, i \in V, \tag{2.5}$$

$$x_{ui} \in \{0, 1\}, \qquad \forall u \in V, \ \forall i \in V, \tag{2.6}$$

$$y_i \in \{0, 1\}, \qquad \forall i \in V. \tag{2.7}$$

Na função objetivo (2.1), minimiza-se o número de anéis a comporem a rede *backbone*, ou seja, o número de dispositivos DCS a serem instalados. As restrições (2.4) estabelecem que cada localidade deve pertencer a exatamente um único anel, e as restrições (2.5) estipulam que se um anel não está ativo nenhuma localidade pode ser atribuída a ele. As restrições (2.2) estabelecem que a demanda total que passará em cada anel deve ser, no máximo, igual a B. A restrição (2.3) estabelece que a capacidade do anel federal deve ser no máximo igual a B. Vale lembrar que o anel federal é encarregado de levar o tráfego entre as localidades pertencentes a anéis diferentes. As restricções (2.6) e (2.7) correspondem às restrições de integralidade.

Essa formulação requer $n^2 + n$ variáveis de decisão e $n^2 + 2n + 1$ restrições. Observe, ainda, que a formulação (P1) apresenta uma dificuldade: as restrições (2.2) e (2.3) são não lineares.

Através de um processo de linearização bastante simples, Goldschmidt *et al.* conseguiram transformar a formulação anterior em um problema de

programação linear inteira 0-1. Considere as novas variáveis de decisão p_{uvi} e z_{uvi} definidas para o problema, tais que $p_{uvi} = x_{ui}x_{vi}$ e $z_{uvi} = x_{ui}(1 - x_{vi})$.

A variável p_{uvi} $(u, v \in V, u < v, i \in V)$ indica se ambas as localidades u e v pertencem ao agrupamento i ($p_{uvi} = 1$), ou $p_{uvi} = 0$ em caso contrário. Essas variáveis devem satisfazer às seguintes desigualdades:

$$p_{uvi} - x_{ui} \leq 0, \quad \forall u, v \in V, u < v, \forall i \in V, \tag{2.8}$$

$$p_{uvi} - x_{vi} \leq 0, \quad \forall u, v \in V, u < v, \forall i \in V, \tag{2.9}$$

$$x_{ui} + x_{vi} - p_{uvi} \leq 1, \quad \forall u, v \in V, u < v, \forall i \in V. \tag{2.10}$$

As desigualdades (2.10) estabelecem que se a localidade u e v pertencem ao agrupamento i, então p_{uvi} será a igual 1.

A segunda variável de decisão z_{uvi} $(u, v \in V, u \neq v, i \in V)$ indica se a localidade u pertence ao agrupamento i e a localidade v não pertence ao agrupamento i ($z_{uvi} = 1$), ou $z_{uvi} = 0$ em caso contrário. Tal variável deve satisfazer as desigualdades abaixo:

$$z_{uvi} - x_{ui} \leq 0, \quad \forall u, v \in V, \forall i \in V, \tag{2.11}$$

$$z_{uvi} + x_{vi} \leq 1, \quad \forall u, v \in V, \forall i \in V, \tag{2.12}$$

$$x_{ui} - x_{vi} - z_{uvi} \leq 0, \quad \forall u, v \in V, \forall i \in V. \tag{2.13}$$

As desigualdades (2.13) estabelecem que se a localidade u pertence ao agrupamento i e a localidade v não pertence a esse agrupamento, então z_{uvi} será a igual 1.

Usando as variáveis de decisão definidas acima, podemos substituir as restrições (2.2) pelas seguintes restrições:

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv}p_{uvi} + \sum_{u=1}^{n} \sum_{v=1}^{n} d_{uv}z_{uvi} \leq B, \quad \forall i \in V, \tag{2.14}$$

e a restrição (2.3) pela restrição abaixo:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv}z_{uvi} \leq B. \tag{2.15}$$

Como desejamos minimizar o número de agrupamentos, e o custo associado a cada variável y_i é maior do que zero, temos que para qualquer solução viável as desigualdades (2.8) e (2.9) serão satisfeitas na otimalidade. Assim, só precisamos trabalhar com a desigualdade (2.10). O mesmo raciocínio pode ser feito com as desigualdades (2.11), (2.12) e (2.13).

Com base nessas novas informações, a formulação (P1) pode ser reescrita como o seguinte problema de programação linear inteira (0-1):

(P1)

$$\text{minimize } \sum_{i=1}^{n} y_i,$$

sujeito a:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv}p_{uvi} + \sum_{u=1}^{n}\sum_{v=1}^{n} d_{uv}z_{uvi} \leq B, \qquad \forall i \in V,$$

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv}z_{uvi} \leq B,$$

$$\sum_{i=1}^{n} x_{ui} = 1, \qquad \forall u \in V,$$

$$x_{ui} - y_i \leq 0, \qquad \forall u, i \in V,$$

$$x_{ui} + x_{vi} - p_{uvi} \leq 1, \qquad \forall u, v \in V, u < v, \ \forall i \in V,$$

$$x_{ui} - x_{vi} - z_{uvi} \leq 0, \qquad \forall u, v \in V, \ \forall i \in V,$$

$$x_{ui} \in \{0,1\}, \qquad \forall u \in V, \ \forall i \in V,$$

$$y_i \in \{0,1\}, \qquad \forall i \in V,$$

$$p_{uvi} \in \{0,1\}, \qquad \forall u, v \in V, u < v, \ \forall i \in V, \tag{2.16}$$

$$z_{uvi} \in \{0,1\}, \qquad \forall u, v \in V, \ \forall i \in V. \tag{2.17}$$

Esta versão da formulação (P1) possui $2mn + mn$ novas variáveis de decisão (devido, respectivamente, a z_{uvi} e a p_{uvi}). Com a linearização das

restrições (2.2) e (2.3), passamos a ter $n^2 + 3mn + n$ variáveis de decisão e $n^2 + 3mn + 2n + 1$ restrições na formulação (P1).

Em seguida, apresentamos e discutimos a segunda formulação inteira 0-1 que propusemos para o problema de atribuição de localidades a anéis.

2.2.2 Formulação de empacotamento

Nesta seção, propomos uma nova formulação linear inteira 0-1 para o problema de atribuição de localidades a anéis. Essa nova formulação, ao invés de possuir uma restrição de particionamento, representada pela restrição (2.4), possuirá uma restrição de empacotamento da forma

$$\sum_{i=1}^{n} x_{ui} \leq 1, \quad \forall u \in V. \tag{2.18}$$

Por isso, a formulação que estamos propondo é denominada formulação de empacotamento.

Essa nova maneira de apresentar o problema é similar à que foi proposta por Martello e Toth [52] para o problema de atribuição generalizado, ou seja, definimos uma variável artificial para cada restrição (2.4). Isto é feito da seguinte forma.

Seja t_u a variável artificial dada por $t_u = 1 - \sum_{i=1}^{n} x_{ui} \geq 0$ para todo $u \in V$. Além disso, seja Θ_u o custo associado a cada variável t_u, onde $\Theta_u > 0$. Assim, a função objetivo (2.1) pode ser reescrita como

$$\overline{z} = \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u t_u. \tag{2.19}$$

Sabendo-se que o custo associado a cada variável de decisão x_{ui} na função objetivo (2.1) é igual a zero, vê-se que esta nova função objetivo, dada por (2.19), está penalizando os vetores x, onde (2.18) é satisfeita como uma desigualdade estrita.

Evidentemente que as variáveis t_u podem ser eliminadas em (2.19). Isso é feito abaixo.

$$\overline{z} = \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u t_u =$$

$$= \min \sum_{i=1}^{n} y_i + \sum_{u=1}^{n} \Theta_u (1 - \sum_{i=1}^{n} x_{ui}) =$$

$$= \min \sum_{i=1}^{n} y_i - \sum_{u=1}^{n} \sum_{i=1}^{n} \Theta_u x_{ui} + \sum_{u=1}^{n} \Theta_u. \tag{2.20}$$

Agora, escrevendo a função objetivo (2.20) como uma função de maximização obtemos a seguinte função para a nova formulação:

$$\overline{z} = \max \sum_{u=1}^{n} \sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i - \sum_{u=1}^{n} \Theta_u. \tag{2.21}$$

Como $\sum_{u=1}^{n} \Theta_u$ é um termo constante na função objetivo (2.21), podemos retirá-lo.

Feitas essas transformações, na função objetivo (2.1) e na restrição (2.4), obtemos a segunda formulação para o problema de atribuição de localidades a anéis. Essa nova formulação pode ser escrita como o problema de programação linear inteira (0-1) logo abaixo:

(P2)

$$\text{maximize} \sum_{u=1}^{n} \sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i,$$

sujeito a:

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} d_{uv} p_{uvi} + \sum_{u=1}^{n} \sum_{v=1}^{n} d_{uv} z_{uvi} \leq B, \qquad \forall i \in V,$$

$$\sum_{u=1}^{n-1} \sum_{v=u+1}^{n} \sum_{i=1}^{n} d_{uv} z_{uvi} \leq B,$$

$$\sum_{i=1}^{n} x_{ui} \leq 1, \qquad \forall u \in V,$$

$$x_{ui} - y_i \leq 0, \qquad \forall u, i \in V,$$

$$
\begin{aligned}
& x_{ui} + x_{vi} - p_{uvi} \leq 1, && \forall u, v \in V, u < v, \ \forall i \in V, \\
& x_{ui} - x_{vi} - z_{uvi} \leq 0, && \forall u, v \in V, \ \forall i \in V \\
& x_{ui} \in \{0,1\}, && \forall u \in V, \ \forall i \in V, \\
& y_i \in \{0,1\}, && \forall i \in V, \\
& p_{uvi} \in \{0,1\}, && \forall u, v \in V, u < v, \ \forall i \in V, \\
& z_{uvi} \in \{0,1\}, && \forall u, v \in V, \ \forall i \in V.
\end{aligned}
$$

O número de variáveis de decisão e restrições da formulação (P2) manteve-se igual ao da formulação (P1). Além disso, pode-se verificar que uma solução inteira (0-1) da formulação (P2) corresponde a uma partição viável do conjunto de localidades, ou seja, agrupamentos que satisfazem as restrições de capacidade. Sendo assim, pode-se deduzir que qualquer solução ótima de (P2) é solução ótima de (P1).

Em seguida, apresentamos um estudo computacional empírico com as duas formulações lineares inteiras $0-1$. Essas formulações foram utilizadas para a resolução de instâncias de pequeno e médio porte do problema de atribuição de localidades a anéis existentes na literatura.

2.3 Resultados computacionais

Nesta seção, apresentamos os resultados computacionais obtidos com as formulações (P1) e (P2), descritas nas seções 2.2.1 e 2.2.2, respectivamente. Esses resultados possibilitaram uma comparação empírica entre as duas formulações. Todos os experimentos foram realizados em um computador com processador `AMD K6 450 MHz` e com `256Mbytes` de memória RAM.

Nos experimentos computacionais foram testadas duas classes de instâncias. Essas instâncias correspondem às mesmas definidas por Goldschmidt *et al.* [26] e por Aringhieri e Dell'Amico [3], nos seus experimentos computacionais. Até o final deste texto, denominaremos por classe C1 e classe C2, respectivamente, as instâncias geradas por Goldschmidt *et al.*, e Aringhieri e Dell'Amico.

Para cada classe de instância, trabalhamos com dois grupos, geométricas

e aleatórias, e com dois tipos de capacidades diferentes para os anéis: $B = 155Mbs$ e $B = 622Mbs$.

Para a resolução das formulações, utilizou-se o pacote de otimização XPRESS 12.05 (vide [14]). Foram estabelecidos dois critérios de parada para o término da execução do procedimento *branch-and-bound*: número máximo de nós na árvore *branch-and-bound* e tempo máximo de CPU. Os valores estipulados foram de 300,000 nós e 18,000 segundos, respectivamente. Para efeito de comparação, a geração automática de planos-de-corte disponível neste pacote foi desabilitada.

Vale ressaltar que o valor da relaxação linear para todas as instâncias testadas foi igual a 1 e que para a formulação (P2) utilizamos $\Theta_u = 1$, para todo $u \in V$.

2.3.1 Classe C1

Essa primeira classe de instâncias, como dito anteriormente, foi gerada por Goldschmidt *et al.* [26]. Fizemos uso de 47 instâncias desta classe para realizar essa primeira fase dos experimentos computacionais com as formulações (P1) e (P2).

Um resumo dessas instâncias é apresentado na Tabela 2.1. A primeira coluna indica o grupo da instância; a segunda, a capacidade máxima que cada anel deve possuir e, por último, a terceira indica a cardinalidade. Nesta última coluna temos, para cada cardinalidade, o número de instâncias que foram testadas.

Grupo	Capacidade (B)	Cardinalidade ($\|V\|$)	
		15	25
geométrica	155Mbs	4	5
geométrica	622Mbs	4	7
aleatória	155Mbs	7	8
aleatória	622Mbs	7	5

Tabela 2.1: Distribuição das instâncias testadas na classe C1 com as formulações (P1) e (P2).

Instâncias geométricas

As instâncias geométricas representam agrupamentos naturais, ou seja, uma localidade tenta se comunicar com outras localidades que estejam mais próximas dela do que com outras que estejam mais distantes. Essa distância corresponde à distância Euclidiana.

Os resultados computacionais para essas instâncias são apresentados nas Tabelas 2.2-2.3. A primeira coluna indica o nome da instância testada. A coluna z^* indica o ótimo inteiro da instância. Para cada formulação, temos as seguintes colunas: `#Nds`, que indica o número total de nós abertos na árvore *branch-and-bound*; z, que indica o melhor limite encontrado pela formulação em questão; Δ, que indica a diferença percentual entre o ótimo inteiro z^* e o limitante z obtido por uma das formulações, isto é:

$$\Delta(z^*, z) = \Delta = \frac{z^* - z}{z^*} \times 100. \tag{2.22}$$

Por último, temos a coluna `Tmp` que indica o tempo de CPU expresso em segundos.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
gl_15_1	3	7208	3	0%	851	4873	3	0%	654
gl_15_4	3	7317	3	0%	823	3994	3	0%	665
gl_15_7	3	7450	3	0%	800	9838	3	0%	1880
gl_15_9	3	300000	2	33.33%	7414	16030	3	0%	1875
gl_25_1	4	16100	1.32	67%	18000	16500	2	50%	18000
gl_25_3	3	32600	1.29	57%	18000	12568	3	0%	11289
gl_25_4	4	15300	1.33	66.75%	18000	10200	2	50%	18000
gl_25_7	3	16600	2	33.33%	18000	12612	3	0%	14755
gl_25_8	4	13200	1.35	66.25%	18000	12400	2	50%	18000

Tabela 2.2: Resultados obtidos com as formulações para as instâncias geométricas da classe C1 com $B = 155Mbs$.

Neste grupo, das 20 instâncias testadas, podemos observar que a formulação (P2) provou a otimalidade de 16 instâncias, enquanto que a formulação (P1) conseguiu provar a otimalidade de apenas 9 instâncias.

Pode-se observar, ainda, que nas instâncias onde foi possível provar a

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
gh_15_1	3	46806	3	0%	2847	11954	3	0%	2222
gh_15_2	3	27695	3	0%	8888	7424	3	0%	2296
gh_15_8	3	7047	3	0%	3053	3645	3	0%	1731
gh_15_9	3	20324	3	0%	3096	12357	3	0%	2599
gh_25_1	3	8600	1.24	58.67%	18000	3202	3	0%	2481
gh_25_2	2	3261	2	0%	1808	923	2	0%	451
gh_25_3	2	2162	2	0%	962	1233	2	0%	689
gh_25_4	3	7000	1.27	57.67%	18000	8466	3	0%	2942
gh_25_7	4	5200	1.32	67%	18000	7100	2	50%	18000
gh_25_8	3	7700	1.27	57.67%	18000	2781	3	0%	4325
gh_25_9	3	7000	1.27	57.67%	18000	7420	3	0%	10261

Tabela 2.3: Resultados obtidos com as formulações para as instâncias geométricas da classe C1 com $B = 622Mbs$.

otimalidade, em apenas uma instância, gl_15_7, a formulação (P2) teve que abrir mais nós na árvore *branch-and-bound* do que a formulação (P1). Isso ocasionou um tempo computacional maior, por parte da formulação (P2), para a resolução desta instância.

Analisemos, agora, os valores obtidos para as instâncias onde a formulação (P2) não conseguiu provar a otimalidade no tempo estipulado. Nessas quatro instâncias, gl_25_1, gl_25_4, gl_25_8 e gh_25_7, nota-se que a formulação (P2) obteve uma diferença percentual sempre menor do que aquela obtida pela formulação (P1).

Vale ressaltar, ainda, que em 75% das instâncias testadas, a formulação (P2) apresentou um tempo computacional inferior ao obtido pela formulação (P1). Por exemplo, para as instâncias gh_25_2 e gh_25_8 esse tempo foi pelo menos duas vezes menor.

Um outro aspecto positivo que podemos ressaltar com relação à formulação (P2) é que para 60% das instâncias, o tempo computacional ficou abaixo de 1 hora, enquanto que para a formulação (P1) essa percentagem foi bem menor, algo em torno de 40%.

Instâncias aleatórias

As instâncias aleatórias foram geradas pelos autores a partir de grafos completos, com a subtração de arestas que tivessem uma probabilidade menor ou igual a uma constante $p \in (0, 1)$ e fixada a priori.

As Tabelas 2.4 e 2.5 apresentam os resultados obtidos para as instâncias aleatórias. Foram testadas 27 instâncias. Com a formulação (P2), provou-se a otimalidade de 20 instâncias, enquanto com que a formulação (P1), isto ocorreu para apenas 15 instâncias.

Observando as instâncias onde foi possível provar a otimalidade, nota-se que em apenas duas instâncias, `rl_15_4` e `rh_15_7`, a formulação (P2) teve que abrir mais nós na árvore *branch-and-bound* do que a formulação (P1). A exemplo das instâncias geométricas, isso ocasionou um tempo computacional maior com a formulação (P2) do que com a formulação (P1) na resolução daquelas instâncias.

Nas instâncias `rl_25_4`, `rl_25_5`, `rl_25_6`, `rl_25_7`, `rl_25_9`, `rl_25_10` e `rh_25_2`, onde não foi possível provar a otimalidade, a formulação (P2) obteve uma diferença percentual menor ou igual àquela que foi obtida pela formulação (P1). Note que o número de nós abertos na árvore *branch-and-bound* para estas instâncias, utilizando a formulação (P2), só não foi menor do que os abertos pela formulação (P1) nas instâncias `rl_25_9` e `rh_25_2`.

Por último, vale ressaltar que em 52% das instâncias, a formulação (P2) apresentou um tempo computacional inferior ao obtido pela formulação (P1). Esta melhoria no tempo computacional pode ser observada, principalmente, através das instâncias `rl_25_2`, `rl_25_8`, `rh_15_1`, `rh_25_3`, `rh_25_7` e `rh_25_10`.

Note, ainda, que para a formulação (P2), o tempo computacional obtido ficou abaixo de 1 hora em 52% das instâncias, enquanto que para a formulação (P1) essa percentagem foi igual a 41%.

2.3.2 Classe C2

As instâncias pertencentes a esta classe foram geradas por Aringhieri e Dell'Amico [3]. Foram testadas 30 instâncias. Um resumo dessas instâncias

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
rl_15_1	3	11671	3	0%	1136	9118	3	0%	1066
rl_15_3	2	170	2	0%	2	168	2	0%	3
rl_15_4	3	5449	3	0%	992	5752	3	0%	1201
rl_15_6	3	14729	3	0%	1245	12548	3	0%	1816
rl_15_8	3	19342	3	0%	2343	9591	3	0%	1775
rl_15_9	3	24793	3	0%	1750	15270	3	0%	1730
rl_15_10	3	11663	3	0%	2306	8853	3	0%	1825
rl_25_2	3	19100	2	33.33%	18000	4869	3	0%	4037
rl_25_4	4	15100	2	50%	18000	9200	2	50%	18000
rl_25_5	4	20700	2	50%	18000	9100	2	50%	18000
rl_25_6	4	14800	2	50%	18000	8200	2	50%	18000
rl_25_7	4	15900	2	50%	18000	7200	2	50%	18000
rl_25_8	3	16900	2	33.33%	18000	4044	3	0%	2136
rl_25_9	4	12800	1.33	66.75%	18000	23100	2	50%	18000
rl_25_10	4	19100	2	50%	18000	12600	2	50%	18000

Tabela 2.4: Resultados obtidos com as formulações para as instâncias aleatórias da classe C1 com $B = 155Mbs$.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
rh_15_1	3	20188	3	0%	4123	4361	3	0%	1450
rh_15_4	2	108	2	0%	3	85	2	0%	6
rh_15_5	3	29568	3	0%	4349	9039	3	0%	2378
rh_15_6	3	37361	3	0%	4335	16800	3	0%	3827
rh_15_7	2	134	2	0%	4	187	2	0%	10
rh_15_8	2	1166	2	0%	26	546	2	0%	34
rh_15_9	3	32864	3	0%	4140	18319	3	0%	3629
rh_25_2	3	8100	1.22	59.33%	18000	15600	2	33.33%	18000
rh_25_3	3	9300	1.23	59%	18000	2354	3	0%	4006
rh_25_7	3	9600	1.22	59.33%	18000	3103	3	0%	4588
rh_25_9	2	374	2	0%	96	353	2	0%	77
rh_25_10	3	10100	2	33.33%	18000	14037	3	0%	13870

Tabela 2.5: Resultados obtidos com as formulações para as instâncias aleatórias da classe C1 com $B = 622Mbs$.

é apresentado na Tabela 2.6.

Grupo	Capacidade (B)	Cardinalidade ($\|V\|$)	
		15	25
geométrica	155Mbs	5	5
geométrica	622Mbs	0	5
aleatória	155Mbs	6	4
aleatória	622Mbs	5	0

Tabela 2.6: Distribuição das instâncias testadas na classe C2 com as formulações (P1) e (P2).

Instâncias geométricas

Nestes experimentos, diferentemente dos apresentados na seção 2.3.1, não conseguimos provar a otimalidade de qualquer uma das 15 instâncias geométricas testadas com ambas as formulações. As Tabelas 2.7 e 2.8 mostram esses resultados.

Em todas as intâncias testadas, o tempo computacional, para ambas as formulações, atingiu o valor máximo estabelecido para os experimentos. Entretanto, em 14 das 15 instâncias, a formulação (P2) obteve limites com uma qualidade melhor, se não igual, à obtida pela formulação (P1). Vale ressaltar, ainda, que essa qualidade no limitante foi obtida com um número bem menor de nós na maioria das instâncias. O que nos leva a acreditar que a formulação (P2) pode ser melhor do que a formulação (P1).

Instâncias aleatórias

Nesse experimento, foram testadas 15 instâncias aleatórias . As Tabelas 2.9 e 2.10 apresentam os resultados obtidos. A formulação (P2) provou a otimalidade de 7 instâncias, enquanto que a formulação (P1), de 8 instâncias. Apesar desta vantagem aparente da formulação (P1), nota-se que, em média, a formulação (P2) enumerou menos nós na árvore *branch-and-bound* para provar a otimalidade daquelas instâncias. Mais uma vez, para estes casos, isto nos leva a crer que a formulação (P2) seria melhor do que a formulação (P1).

Observemos, agora, os valores obtidos para as instâncias onde não foi possível provar a otimalidade. Nessas instâncias, a formulação (P2) obteve uma diferença percentual menor ou igual àquela obtida pela formulação (P1) em todas essas instâncias.

Por último, vale ressaltar que em 26.67% das instâncias, a formulação (P2) apresentou um tempo computacional inferior ao obtido pela formulação (P1). Em particular, ambas as formulações ficaram com um tempo computacional abaixo de 1 hora em 26.67% das instâncias.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
gl_15_3.1	4	65800	3	25%	18000	40000	2.32	42%	18000
gl_15_3.2	4	59400	2.19	45.25%	18000	37500	2.31	42.25%	18000
gl_15_3.3	4	69800	2.20	45%	18000	37200	2.30	42.50%	18000
gl_15_3.4	4	58100	2.17	45.75%	18000	33500	2.29	42.75%	18000
gl_15_3.5	4	53400	2.16	46%	18000	37500	2.30	42.50%	18000
gl_25_9.1	4	14300	2	50%	18000	16600	2	50%	18000
gl_25_9.2	4	18400	2	50%	18000	14800	2	50%	18000
gl_25_9.4	4	10800	2	50%	18000	10500	2	50%	18000
gl_25_10.4	5	4200	1.37	72.60%	18000	20300	2	50%	18000
gl_25_10.5	5	3900	1.35	73%	18000	26300	2	50%	18000

Tabela 2.7: Resultados obtidos com as formulações para as instâncias geométricas da classe C2 com $B = 155Mbs$.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
gh_25_5.2	4	6100	1.29	67.75%	18000	4800	1.35	66.25%	18000
gh_25_5.3	4	2600	1.29	67.75%	18000	3200	1.41	64.75%	18000
gh_25_5.4	4	1800	1.30	67.50%	18000	4400	1.40	65%	18000
gh_25_5.5	4	2600	1.30	67.50%	18000	4200	1.41	64.75%	18000
gh_25_5.6	4	8000	1.29	67.75%	18000	2900	1.47	63.25%	18000

Tabela 2.8: Resultados obtidos com as formulações para as instâncias geométricas da classe C2 com $B = 622Mbs$.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
rl_15_2.1	3	10286	3	0%	2143	5783	3	0%	858
rl_15_2.2	4	56300	2.09	47.75%	18000	114900	2.37	40.75%	18000
rl_15_2.3	3	28293	3	0%	4513	9514	3	0%	1221
rl_15_5.1	3	12619	3	0%	2351	17745	3	0%	6673
rl_15_5.2	3	7329	3	0%	2128	15193	3	0%	4803
rl_15_5.3	3	5782	3	0%	1822	3009	3	0%	981
rl_25_3.1	4	17700	1.33	66.75%	18000	10700	2	50%	18000
rl_25_3.2	4	15500	1.32	67%	18000	5400	2	50%	18000
rl_25_3.3	4	14300	1.31	67.25%	18000	17000	1.46	63.50%	18000
rl_25_3.5	4	17900	1.33	66.75%	18000	12200	2	50%	18000

Tabela 2.9: Resultados obtidos com as formulações para as instâncias aleatórias da classe C2 com $B = 155Mbs$.

		Formulação (P1)				Formulação (P2)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
rh_15_10.1	3	11190	3	0%	6188	11200	2	33.33%	18000
rh_15_10.2	3	17964	3	0%	11246	3421	3	0%	4113
rh_15_10.4	3	30100	2	33.33%	18000	11100	2	33.33%	18000
rh_15_10.6	3	10064	3	0%	5826	12533	3	0%	15542
rh_15_10.8	3	31200	2	33.33%	18000	11200	2	33.33%	18000

Tabela 2.10: Resultados obtidos com as formulações para as instâncias aleatórias da classe C2 com $B = 622Mbs$.

2.4 Considerações

Os resultados computacionais obtidos através dos experimentos feitos com as duas formulações indicaram que a formulação (P2), que propusemos, mostrou-se melhor do que a formulação (P1) em vários aspectos:

- das 77 instâncias testadas, a formulação (P2) provou a otimalidade em 43 delas, enquanto que a formulação (P1) provou a otimalidade em apenas 32 instâncias;
- em 28 das 77 instâncias, a formulação (P2) obteve uma diferença percentual entre 1% e 50%. Já a formulação (P1) obteve estes valores para a diferença percentual em 21 instâncias;
- o número de nós abertos na árvore *branch-and-bound*, utilizando a formulação (P2), foi inferior ao da formulação (P1) em 55 instâncias;
- o tempo computacional obtido pela formulação (P2) foi menor do que o obtido pela formulação (P1) em 33 instâncias, o que pode ser considerado notável, já que em 33 outras instâncias, ambas as formulações tiveram suas execuções interrompidas pelo limite máximo de tempo permitido.

No capítulo seguinte, faremos um estudo poliedral da formulação (P2), objetivando melhorar ainda mais esta formulação.

Capítulo 3

Estudo poliédrico do problema de atribuição de localidades a anéis

Um dos principais objetivos deste trabalho consiste em estudar e comparar diferentes formulações do problema de atribuição de localidades a anéis como um problema de programação linear inteira. Parte deste objetivo foi realizado no capítulo anterior. Neste capítulo, apresentamos um estudo sobre a estrutura facial do politopo associado ao SRAP.

Através da formulação de empacotamento, foi possível definirmos novas classes de desigualdades válidas para o politopo associado ao problema de atribuição de localidades a anéis. São apresentadas, também, condições para que algumas desigualdades sejam definidoras de facetas.

Em seguida, apresentamos um estudo sobre a simetria das soluções do problema de atribuição de localidades a anéis. Uma das maneiras empregadas na literatura para reduzir esta dificuldade é através do emprego de desigualdades lineares. Com base neste estudo, definimos algumas desigualdades para quebrar essa simetria das soluções.

3.1 Resultados para o poliedro do SRAP

Nesta seção, será feito um estudo poliedral do problema de atribuição de localidades a anéis. Como no capítulo anterior, a instância do problema será dada por um grafo não-dirigido $G = (V, E)$, com $|V| = n$ e $|E| = m$, um inteiro positivo B, denotando a capacidade de cada agrupamento, e um vetor

de demandas d, que associa valores inteiros não-negativos a cada aresta de E.

Conforme explicado no Capítulo 2, optamos por estudar a formulação (P2), que é reproduzida abaixo:

(P2)

$$\text{maximize } \sum_{u=1}^{n}\sum_{i=1}^{n}\Theta_u x_{ui} - \sum_{i=1}^{n} y_i, \tag{2.21}$$

sujeito a:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv}p_{uvi} + \sum_{u=1}^{n}\sum_{v=1}^{n} d_{uv}z_{uvi} \leq B, \qquad \forall i \in V, \tag{2.14}$$

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv}z_{uvi} \leq B, \tag{2.15}$$

$$\sum_{i=1}^{n} x_{ui} \leq 1, \qquad \forall u \in V, \tag{2.18}$$

$$x_{ui} - y_i \leq 0, \qquad \forall u, i \in V, \tag{2.5}$$

$$x_{ui} + x_{vi} - p_{uvi} \leq 1, \qquad \forall u, v \in V, u < v, \forall i \in V, \tag{2.10}$$

$$x_{ui} - x_{vi} - z_{uvi} \leq 0, \qquad \forall u, v \in V, \forall i \in V, \tag{2.13}$$

$$x_{ui} \in \{0,1\}, \qquad \forall u \in V,\ \forall i \in V, \tag{2.6}$$

$$y_i \in \{0,1\}, \qquad \forall i \in V, \tag{2.7}$$

$$p_{uvi} \in \{0,1\}, \qquad \forall u, v \in V, u < v,\ \forall i \in V, \tag{2.16}$$

$$z_{uvi} \in \{0,1\}, \qquad \forall u, v \in V,\ \forall i \in V. \tag{2.17}$$

Nesta formulação, trabalhamos com as variáveis de decisão p_{uvi} e z_{uvi}. Estas variáveis são definidas apenas sobre as arestas, ou seja, apenas quando tivermos $d_{uv} > 0$.

O politopo associado à formulação (P2) é dado pela envoltória convexa de suas soluções inteiras, ou seja:

$$\Omega = \ conv\,\{(x,y,p,z) \in \mathbb{B}^{n^2+n+mn+2mn} \mid (x,y,p,z) \text{ satisfaz } (2.14), (2.15), (2.18), (2.5), (2.10), (2.13), (2.6), (2.7), (2.16), (2.17)\}.$$

Nas subseções seguintes, estudaremos a dimensão e algumas desigualdades válidas para o politopo $\mathcal{Q}$. Antes, porém, introduzimos alguma notação auxiliar. O vetor nulo em um espaço de dimensão k será denotado por 0^k, enquanto que e_i^k será usado para denotar o vetor deste espaço com todas as componentes nulas, excetuando-se a i-ésima componente que possui valor 1. Quando o espaço no qual estivermos trabalhando admitir uma partição natural das componentes dos seus vetores, mais precisamente, quando o espaço tiver dimensão $k \times t$, usaremos a notação $e_{ij}^{k\times t}$ para nos referirmos ao vetor em que todas as componentes são nulas, excetuando-se a $(k \times i + j)$-ésima componente que vale 1.

3.1.1 Resultados básicos sobre o politopo $\mathcal{Q}$

Nesta seção, serão estudados alguns resultados básicos sobre o politopo $\mathcal{Q}$, como a dimensão e a força das desigualdades presentes na formulação (P2), ou seja, procuraremos, sempre que possível, caracterizar quais dentre estas desigualdades são definidoras de facetas para o referido politopo. Será assumido de agora em diante que $d_{uv} \leq B$ para todo u e v em V.

Teorema 3.1 *Para $n \geq 2$, a dimensão do politopo $\mathcal{Q}$ é cheia.*

Prova: Os seguintes vetores de $\mathbb{B}^{n^2+n+mn+2mn}$ pertencem a $\mathcal{Q}$ e são claramente afim independentes:

- o vetor nulo;
- os vetores $(e_{ui}^{n^2}, e_i^n, 0^{mn}, 0^{2mn})$ para todo u e i em V;
- os vetores $(0^{n^2}, e_i^n, 0^{mn}, 0^{2mn})$ para todo $i \in V$;
- os vetores $(0^{n^2}, 0^n, e_{uvi}^{mn}, 0^{2mn})$ para todo u, v e i em V, com $u < v$;
- os vetores $(0^{n^2}, 0^n, 0^{mn}, e_{uvi}^{2mn})$ para todo u, v e i em V, com $u \neq v$.

Portanto, $\mathcal{Q}$ contém $n^2 + n + mn + 2mn$ vetores afim independentes e o resultado está provado.

□

Como o politopo $\mathcal{Q} \subseteq \mathbb{B}^{n^2+n+mn+2mn}$ possui dimensão cheia para $n \geq 2$, então cada faceta de $\mathcal{Q}$ é definida de forma única, a menos de uma multiplicação por um escalar positivo.

Até o final desse texto, empregaremos a técnica de verificação da maximalidade na prova das desigualdades definidoras de facetas. Tal técnica consiste em provar que a face $\mathcal{F}$, definida por uma desigualdade válida para o politopo $\mathcal{Q}$, não está contida em qualquer outra face (*maximalidade*) de $\mathcal{Q}$. Abaixo, segue uma breve explicação sobre o uso desta técnica de demonstração.

Seja $\mathcal{F} = \{x \in \mathcal{Q} : \pi x = \pi_0\}$. Assuma que $\gamma x \leq \gamma_0$ é uma desigualdade válida qualquer para o politopo $\mathcal{Q}$, tal que $\mathcal{F} \subseteq \{x \in \mathcal{Q} : \gamma x = \gamma_0\}$. Se $\gamma x \leq \gamma_0$ pode ser expresso como uma combinação linear da desigualdade $\pi x \leq \pi_0$ multiplicada por um escalar positivo, ou seja, $(\gamma, \gamma_0) = (\alpha\pi, \alpha\pi_0)$ para algum $\alpha \in \mathbb{R}_+$, então $\pi x \leq \pi_0$ é uma faceta para o politopo $\mathcal{Q}$.

Feita esta explanação sobre a técnica de maximalidade, retornemos à caracterização da estrutura facial do politopo $\mathcal{Q}$. A relevância deste estudo deve-se à inexistência de um trabalho semelhante na literatura, isto é, que faça uma investigação poliedral do SRAP.

O nosso estudo sobre o politopo $\mathcal{Q}$ começa com as desigualdades da formulação inteira (P2). Procuraremos caracterizar quais destas desigualdades são definidoras de facetas para o politopo $\mathcal{Q}$. Nas seções seguintes, apresentaremos outras desigualdades válidas e definidoras de facetas para $\mathcal{Q}$.

Nas provas de facetas, adotaremos a seguinte convenção. Os agrupamentos serão caracterizados pelos conjuntos $V_1, \ldots, V_n$. Soluções viáveis para a formulação (P2) serão dadas por um vetor da forma $\chi^{V_{j(1)},\ldots,V_{j(q)}}_{U_1,U_2,\ldots,U_q} = (x, y, p, z)$ em $\mathbb{B}^{n^2+n+mn+2mn}$, onde $j(i) \in \{1, \ldots, n\}$ para todo $i = 1, \ldots, q$. Nesta solução, o agrupamento $V_{j(i)}$ está ativo e os vértices do conjunto U_i, se este não for vazio, são alocados a este agrupamento. No entanto, a formulação (P2) pode deixar algumas variáveis dos vetores p e z livres para assumirem os valores 0 ou 1, mesmo quando x e y estão fixados. Assim, para que este vetor fique bem definido, convencionaremos que $\chi^{V_{j(1)},\ldots,V_{j(q)}}_{U_1,U_2,\ldots,U_q} = (x, y, p, z)$ é dado por:

- $x_{u,j(i)} = 1$, se e somente se $u \in U_i$;

- $y_j = 1$, se e somente se existe $i \in \{1, \ldots, q\}$, tal que $j = j(i)$;
- $p_{u,v,j(i)} = 1$, se e somente se $u \in U_i$ e $v \in U_i$;
- $z_{u,v,j(i)} = 1$, se e somente se $u \in U_i$ e $v \in U_j$ para algum j pertencente a $\{1, \ldots, q\} \setminus \{i\}$.

Para simplificar a notação, a lista de agrupamentos ficará restrita àqueles agrupamentos ativos. Além disso, os conjuntos U_i serão representados simplesmente por um índice, quando este se referir ao único vértice de U_i. Finalmente, se apenas um conjunto U_i for não vazio, ele será representado por uma seqüência de índices separados por vírgulas; se todos os conjuntos U_i forem vazios, a lista de conjuntos será omitida. Exemplos de uso da notação que adotamos são dados abaixo:

- $\mathcal{X}^{V_i}$ significa que todas as componentes do vetor são nulas, exceto a componente y_i que vale 1;
- $\mathcal{X}^{V_j,V_i}_{\{r,s\},u}$ significa que $x_{rj} = x_{sj} = x_{ui} = y_j = y_i = p_{rsj} = z_{uri} = z_{usi} = z_{ruj} = z_{suj} = 1$,e todas as demais componentes do vetor são nulas.
- $\mathcal{X}^{V_i}_{u,v}$ significa que $x_{ui} = x_{vi} = y_i = p_{uvi} = 1$, e todas as demais componentes do vetor são nulas.
- $\mathcal{X}^{V_i}_{u}$ significa que $x_{ui} = y_i = 1$, e todas as demais componentes do vetor são nulas.

Em seguida, caracterizamos quais desigualdades da formulação (P2) são definidoras de facetas para o politopo $\mathcal{Q}$.

Teorema 3.2 *Se $n \geq 2$, então cada desigualdade*

$$\sum_{i=1}^{n} x_{ui} \leq 1, \quad \forall u \in V, \tag{2.18}$$

define uma faceta para o politopo $\mathcal{Q}$.

Prova:

Assuma que $\mathcal{F}$ seja a face expressa pela desigualdade (2.18) e que $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ uma desigualdade que define uma face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$ para o politopo Ω, tal que $\mathcal{F} \subseteq \mathcal{F}_{(\alpha,\beta,\delta,\mu)} = \{(x, y, p, z) \in \Omega : \alpha x + \beta y + \delta p + \mu z = \pi_0\}$.

Inicialmente, provaremos que a face $\mathcal{F}$ é própria e, em seguida, deduziremos os múltiplos escalares positivos. Como o vetor $\chi_u^{V_i}$ está em $\mathcal{F}$, $\mathcal{F}$ não é uma face vazia. Pode-se observar, também, que o vetor nulo está em $\Omega \setminus \mathcal{F}$. Logo, a face $\mathcal{F}$ é própria.

Após provarmos que a face $\mathcal{F}$ definida pela desigualdade (2.18) é própria, deduziremos, em seguida, os múltiplos escalares positivos para finalizarmos a prova deste teorema. Considere os casos abaixo:

Caso 1) $u \in V_i$ para algum $i \in V$.

Neste caso, temos $x_{ui} = y_i = 1$. Logo, segue-se que $\chi_u^{V_i} \in \mathcal{F}$. Com isso, obtemos $\alpha_{ui} + \beta_i = \pi_0$.

Caso 2) Considere que o agrupamento V_j, com $j \neq i$, esteja ativo e $u \in V_i$.

Neste caso, temos $x_{ui} = y_i = y_j = 1$. Isto implica em $\chi_u^{V_i,V_j} \in \mathcal{F}$. Logo, $\alpha_{ui} + \beta_i + \beta_j = \pi_0$. Como $\alpha_{ui} + \beta_i = \pi_0$, devido ao caso (1), pode-se concluir que $\beta_j = 0$.

Trocando-se os índices de i e j nos casos (1) e (2), é possível concluirmos que $\beta_j = 0$ para todo $j \in V$.

Caso 3) $u \in V_i$ e $v \in V_j$, com $(v, j) \neq (u, i)$.

Neste caso, temos $x_{ui} = x_{vj} = y_i = y_j = 1$. Segue-se que $\chi_{u,v}^{V_i,V_j} \in \mathcal{F}$. Portanto, $\alpha_{ui} + \alpha_{vj} + \beta_i + \beta_j = \pi_0$. Através dos casos (1)-(2) podemos concluir que $\alpha_{vj} = 0$.

Caso 4) $u \in V_i$ e $r, v \in V_j$, com r, v e j quaisquer e algum $i \in V$.

Neste caso, temos $x_{ui} = x_{rj} = x_{vj} = p_{rvj} = y_i = y_j = 1$. Isto implica em $\chi_{u,r,v}^{V_i,V_j} \in \mathcal{F}$. Logo, $\alpha_{ui} + \alpha_{rj} + \alpha_{vj} + \delta_{rvj} + \beta_i + \beta_j = \pi_0$. Sabendo-se que $\alpha_{ui} + \beta_i = \pi_0$, e que $\alpha_{rj} = \alpha_{vj} = \beta_j = 0$, obtemos $\delta_{rvj} = 0$.

Caso 5) $u \in V_i$, $r \in V_j$ e $v \notin V_j$, com r, v e j quaisquer e algum $i \in V$.

Seja V_l, com $l \neq i$, o agrupamento ao qual o vértice v pertence, ou seja, $v \in V_l$. Neste caso, obtemos $x_{ui} = x_{rj} = x_{vl} = z_{rvj} = y_i = y_j = y_l = 1$. Segue-se que $\mathcal{X}_{u,r,v}^{V_i,V_j,V_l} \in \mathcal{F}$. Portanto, $\alpha_{ui} + \alpha_{rj} + \alpha_{vl} + \beta_i + \beta_j + \beta_l + \mu_{rvj} = \pi_0$. Sabendo-se que $\alpha_i + \beta_i = \pi_0$, $\alpha_{rj} = \alpha_{vl} = 0$ e que $\beta_j = \beta_l = 0$, através dos casos (1), (2) e (3) , respectivamente, podemos concluir que $\mu_{rvj} = 0$.

Caso 6) $u, v \in V_i$, com $u \neq v$.

Neste caso, temos $x_{ui} = x_{vi} = p_{uvi} = y_i = 1$. Logo, segue-se que $\mathcal{X}_{u,v}^{V_i} \in \mathcal{F}$. Portanto, $\alpha_{ui} + \alpha_{vi} + \beta_i + \delta_{uvi} = \pi_0$. Sabemos que $\alpha_{ui} + \beta_i = \pi_0$ e $\delta_{uvi} = 0$, através dos casos (1) e (4), respectivamente. Logo, temos que $\alpha_{vi} = 0$.

Até o momento, temos os seguintes valores:

- $\pi_0 = 0$ e $\beta_j = 0$ para todo $j \in V$;
- $\alpha_{rj} = 0$ para todo $r, j \in V$, com $(r, j) \neq (u, i)$;
- $\delta_{rsj} = 0$ para todo $r, s, j \in V$, com $r < s$;
- $\mu_{rsj} = 0$ para todo $r, s, j \in V$, com $r \neq s$.

Falta calcularmos os coeficientes de α_{ui} e β_i.

Caso 7) Assuma que os vértices u e v possam ser atribuídos a um agrupamento i qualquer, tal que tenhamos uma solução viável. Logo, temos que $\alpha_{ui} = \pi_0$, devido aos casos (1), (2) e (6). Variando o valor de i no caso (1), obtemos $\alpha_{ui} = \pi_0$ para todo $i \in V$.

Assim, concluímos a prova, uma vez que $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ pode ser escrita a partir da desigualdade (2.18) e dos múltiplos escalares positivos encontrados nos casos (1)-(7).

□

Teorema 3.3 *Se $n \geq 2$, então cada desigualdade*

$$x_{ui} - y_i \leq 0, \quad \forall u \in V, \quad \forall i \in V, \tag{2.5}$$

define uma faceta para o politopo Ω.

Prova:

Seja $\mathcal{F}$ a face expressa pela desigualdade (2.5) e $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ uma desigualdade que define uma face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$ para o politopo $\mathcal{Q}$, tal que $\mathcal{F} \subseteq \mathcal{F}_{(\alpha,\beta,\delta,\mu)} = \{(x, y, p, z) \in \mathcal{Q} : \alpha x + \beta y + \delta p + \mu z = \pi_0\}$.

Note-se que $\mathcal{F} \neq \emptyset$, visto que o vetor nulo está em $\mathcal{F}$. Pode-se observar, ainda, que $\mathcal{F} \neq \mathcal{Q}$, uma vez que o vetor $(0^{n^2}, e_j^n, 0^{mn}, 0^{2mn})$ está em $\mathcal{Q} \setminus \mathcal{F}$ para todo $j \in V \setminus \{i\}$. Portanto, temos que $\mathcal{F}$ é uma face própria.

Assuma agora os casos abaixo, para finalizarmos a prova do teorema:

Caso 1) Como o vetor nulo está em $\mathcal{F}$, segue que $\pi_0 = 0$.

Caso 2) Considere que o agrupamento V_j, com $j \neq i$, esteja ativo.

Neste caso, temos $y_j = 1$. Como $\mathcal{X}^{V_j} \in \mathcal{F}$, segue-se que $\beta_j = \pi_0 = 0$.

Caso 3) $r \in V_j$, com $(r, j) \neq (u, i)$.

Sabemos que $x_{rj} = y_j = 1$. Através do vetor de incidência $\mathcal{X}_r^{V_j} \in \mathcal{F}$, obtemos $\alpha_{rj} + \beta_j = \pi_0 = 0$. Como $\beta_j = 0$, pode-se concluir que $\alpha_{rj} = 0$.

Caso 4) $r, s \in V_j$, com $(r, s, j) \neq (u, v, i)$.

Neste caso, temos $x_{rj} = x_{sj} = p_{rsj} = y_j = 1$, ou seja, $\mathcal{X}_{r,s}^{V_j} \in \mathcal{F}$. Logo, $\alpha_{rj} + \alpha_{sj} + \beta_j + \delta_{rsj} = \pi_0 = 0$. Como $\alpha_{rj} = \alpha_{sj} = \beta_j = 0$, pelos casos (2)-(3), conclui-se que $\delta_{rsj} = 0$.

Caso 5) $r \in V_j$ e $s \in V_l$, com $l \neq j$ e $(r, s) \neq (u, v)$.

Neste caso, temos $x_{rj} = x_{sl} = z_{rsj} = y_j = y_l = 1$. Isto implica que $\mathcal{X}_{r,s}^{V_j,V_l} \in \mathcal{F}$. Logo, $\alpha_{rj} + \alpha_{sl} + \beta_j + \beta_l + \mu_{rsj} = \pi_0 = 0$. Sabendo-se que $\alpha_{rj} = \alpha_{sl} = \beta_j = \beta_l = 0$, através dos casos (2)-(3), obtemos $\mu_{rsj} = 0$.

Até o momento, temos:

- $\pi_0 = 0$ e $\beta_j = 0$ para todo $j \in V$, com $j \neq i$;
- $\alpha_{rj} = 0$ para todo $r, j \in V$, com $(r, j) \neq (u, i)$;
- $\delta_{rsj} = 0$ para todo $r, s, j \in V$, com $(r, s, j) \neq (u, v, i)$ e $r < s$;
- $\mu_{rsj} = 0$ para todo $r, s, j \in V$, com $(r, s, j) \neq (u, v, i)$ e $r \neq s$.

Falta calcularmos os coeficientes de α_{ui} e β_i.

Caso 6) $u \in V_i$.

Com isso, obtemos $x_{ui} = y_i = 1$. Segue-se que $\mathcal{X}_u^{V_i} \in \mathcal{F}$. Logo, $\alpha_{ui} + \beta_i = \pi_0 = 0$, ou seja, $\alpha_{ui} = -\beta_i = \theta$.

Portanto, a desigualdade $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ é igual a $\theta \times (x_{ui} - y_i \leq 0)$ através dos múltiplos escalares positivos encontrados nos casos (1)-(6). Logo, a prova está concluída.

□

Teorema 3.4 *Se* $n \geq 2$, *então cada desigualdade*

$$x_{ui} - x_{vi} - z_{uvi} \leq 0, \qquad \forall u, v \in V, \ \forall i \in V, \tag{2.13}$$

define uma faceta para o politopo $\mathcal{Q}$.

Prova:

Seja $\mathcal{F} = \{(x, y, p, z) \in \mathcal{Q} : x_{ui} - x_{vi} - z_{uvi} = 0\}$ uma face para o politopo $\mathcal{Q}$. Suponha que $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ seja uma desigualdade válida e que define uma face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$ para o politopo $\mathcal{Q}$, tal que $\mathcal{F} \subseteq \mathcal{F}_{(\alpha,\beta,\delta,\mu)} = \{(x, y, p, z) \in \mathcal{Q} : \alpha x + \beta y + \delta p + \mu z = \pi_0\}$.

Sabemos que o vetor nulo está em $\mathcal{F}$ e, portanto, $\mathcal{F} \neq \emptyset$. Por outro lado, o vetor $\mathcal{X} = (0^{n^2}, 0^n, 0^{mn}, e_{uvi}^{2mn})$ está em $\mathcal{Q} \setminus \mathcal{F}$. Conseqüentemente, temos que a face $\mathcal{F}$ é uma face própria de $\mathcal{Q}$.

Desta forma, poderemos concluir que (2.13) define uma faceta para o politopo $\mathcal{Q}$, se $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ for múltiplo escalar não-negativo da desigualdade (2.13).

Assuma agora os casos abaixo, para finalizarmos a prova:

Caso 1) Como o vetor nulo está em $\mathcal{F}$, segue-se que $\pi_0 = 0$.

Caso 2) Considere que o agrupamento V_j, com $j \in \{1, \ldots, n\}$, esteja ativo e os demais agrupamentos inativos. Neste caso, teremos $y_j = 1$. Como $\mathcal{X}^{V_j} \in \mathcal{F}$, segue-se que $\beta_j = \pi_0 = 0$.

Caso 3) $r \in V_j$, com $(r,j) \neq (u,i)$ e $(r,j) \neq (v,i)$.

Nota-se que $x_{rj} = y_j = 1$, logo $\mathcal{X}_r^{V_j} \in \mathcal{F}$. Isto implica em $\alpha_{rj} + \beta_j = \pi_0$. Através dos casos (1) e (2), podemos deduzir $\alpha_{rj} = 0$.

Caso 4) Para toda tripla (r,s,j) em $V \times V \times V$, o vetor $\mathcal{X}$, com todas as componentes nulas, excetuando-se a componente p_{rsj}, cujo valor é 1, está em $\mathcal{F}$. Com isto, chega-se a $\delta_{rsj} = 0$ para todas as arestas do grafo e todos os agrupamentos.

Caso 5)

Analogamente, excetuando-se o caso onde $(r,s,j) = (u,v,i)$, o vetor $\mathcal{X}$ com todas as componentes nulas, excetuando-se a componente z_{rsj}, cujo valor é 1 está em $\mathcal{F}$. Isto implica que $\mu_{rsj} = 0$ para toda aresta do grafo e todo agrupamento, à exceção da aresta (u,v), quando o agrupamento for i.

Até o momento, sabemos que :

- $\pi_0 = 0$ e $\beta_j = 0$ para todo $j \in V$, com $j \neq i$;
- $\alpha_{rj} = 0$ para todo $r, j \in V$, com $(r,j) \neq (u,i)$;
- $\delta_{rsj} = 0$ para todo $r, s, j \in V$, com $r < s$;
- $\mu_{rsj} = 0$ para todo $r, s, j \in V$, com $(r,s,j) \neq (u,v,i)$ e $r \neq s$.

Falta calcularmos os coeficientes de α_{ui}, α_{vi} e μ_{uvi}.

Caso 6) $u, v \in V_i$.

Neste caso, temos $x_{ui} = x_{vi} = p_{uvi} = y_i = 1$. Uma vez que $\mathcal{X}_{u,v}^{V_i} \in \mathcal{F}$, temos $\alpha_{ui} + \alpha_{vi} + \beta_i + \delta_{uvi} = \pi_0$. Pode-se concluir que $\alpha_{ui} = -\alpha_{vi} = \theta$, visto que $\pi_0 = \beta_i = \delta_{uvi} = 0$, através dos casos (1), (2) e (4), respectivamente.

Caso 7) $u \in V_i$ e $v \notin V_i$.

Assuma que $v \in V_j$ com $j \neq i$. Neste caso, temos $x_{ui} = x_{vj} = z_{uvi} = z_{vuj} = y_i = y_j = 1$. Isto implica em $\mathcal{X}_{u,v}^{V_i,V_j} \in \mathcal{F}$. Logo, $\alpha_{ui} + \alpha_{vj} + \beta_i + \beta_j + \mu_{uvi} + \mu_{vuj} = \pi_0$. Segue-se que $\alpha_{ui} = -\mu_{uvi} = \theta$, pois $\alpha_{vj} = \beta_i = \beta_j = \mu_{vuj} = \pi_0 = 0$, devido aos casos (1), (2), (3) e (5), respectivamente.

Logo, a desigualdade $\alpha x+\beta y+\delta p+\mu z \leq \pi_0$ é igual a $\theta \times (x_{ui} - x_{vi} - z_{uvi} \leq 0)$, através dos múltiplos escalares positivos encontrados nos casos (1)-(7). Assim, a prova está concluída.

□

Considere a desigualdade (2.10) e a face $\mathcal{F}$ definida por ela no politopo $\mathcal{Q}$, ou seja, $\mathcal{F} = \{(x, y, p, z) \in \mathcal{Q} \ : \ x_{ui} + x_{vi} - p_{uvi} = 1\}$. É fácil observar que todos os pontos sobre a face $\mathcal{F}$ satisfazem $y_i = 1$ e, portanto, $\mathcal{F}$ não define uma faceta para o politopo $\mathcal{Q}$. No entanto, é possível realizarmos uma operação de *lifting* e estabelecermos condições para que a desigualdade $x_{ui} + x_{vi} - p_{uvi} \leq 1$ defina uma faceta para o politopo $\mathcal{Q}$. Isto é feito da seguinte forma.

Seja $\Delta \in \mathbb{R}_+$, tal que a desigualdade $\Delta(1 - y_i) + x_{ui} + x_{vi} - p_{uvi} \leq 1$ é válida para $\mathcal{Q}$. O objetivo é encontrar o maior valor possível para Δ (veja [59] para detalhes sobre *lifting*). Quando $y_i = 1$, a desigualdade $\Delta(1 - y_i) + x_{ui} + x_{vi} - p_{uvi} \leq 1$ reduz-se ao caso original e, assim, é trivialmente válida para o politopo $\mathcal{Q}$. Assuma, então, que $y_i = 0$. Neste caso, obtemos $x_{ui} = x_{vi} = p_{uvi} = 0$ e, conseqüentemente, o maior valor de Δ, tal que $y_i = 0$, deve ser igual a 1. Com isto, obtemos a seguinte desigualdade:

$$x_{ui} + x_{vi} - p_{uvi} - y_i \leq 0. \tag{3.1}$$

A face definida no politopo $\mathcal{Q}$ pela desigualdade (3.1) contém todos os pontos de $\mathcal{F}$ e pelo menos mais um ponto afim independente, dado pelo vetor nulo. Abaixo provamos que, de fato, esta face de dimensão maior é uma faceta do politopo $\mathcal{Q}$. Isto significa que as retrições (2.10) podem ser removidas da formulação (P2) e substituídas pelas restrições (3.1).

Teorema 3.5 *Para $n \geq 2$, a desigualdade*

$$x_{ui} + x_{vi} - p_{uvi} - y_i \leq 0, \quad \forall u, v \in V, \ u < v, \quad \forall i \in V,$$

define uma faceta para o politopo $\mathcal{Q}$.

Prova:

Seja $\mathcal{F}$ uma face expressa pela desigualdade (3.1) e $\alpha x+\beta y+\delta p+\mu z \leq \pi_0$ uma desigualdade que define uma face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$ para o politopo $\mathcal{Q}$, tal que $\mathcal{F} \subseteq \mathcal{F}_{(\alpha,\beta,\delta,\mu)} = \{(x,y,p,z) \in \mathcal{Q} : \alpha x + \beta y + \delta p + \mu z = \pi_0\}$.

Os argumentos que antecedem o enunciado deste teorema mostram que a desigualdade (3.1) é válida. Assim, devemos provar inicialmente que a face $\mathcal{F}$ é uma face própria. Isto pode ser feito facilmente, observando que o vetor nulo está em $\mathcal{F}$ e que o vetor $(0^{n^2}, 0^n, e^{mn}_{uvi}, 0^{2mn})$ está em $\mathcal{Q} \setminus \mathcal{F}$.

Para finalizarmos a prova do teorema, considere os casos abaixo:

Caso 1) Como o vetor nulo está em $\mathcal{F}$, segue-se que $\pi_0 = 0$.

Caso 2) Assuma que o agrupamento V_j, com $j \neq i$, esteja ativo.

Neste caso, teremos $y_j = 1$. Como $\mathcal{X}^{V_j} \in \mathcal{F}$, segue-se que $\beta_j = \pi_0 = 0$.

Caso 3) $r \in V_j$, com $(r,j) \neq (u,i)$.

Sabemos que $x_{rj} = y_j = 1$, logo $\mathcal{X}^{V_j}_r \in \mathcal{F}$. Isto implica em $\alpha_{rj} + \beta_j = 0$. Como $\beta_j = 0$, devido ao caso (2), e $\pi_0 = 0$, devido ao caso (1), pode-se concluir que $\alpha_{rj} = 0$.

Caso 4) $r, s \in V_j$, com $(r,s,j) \neq (u,v,i)$.

Neste caso, temos $x_{rj} = x_{sj} = p_{rsj} = y_j = 1$, ou seja, $\mathcal{X}^{V_j}_{r,s} \in \mathcal{F}$. Logo, $\alpha_{rj} + \alpha_{sj} + \beta_j + \delta_{rsj} = \pi_0$. Como $\alpha_{rj} = \alpha_{sj} = \beta_j = \pi_0 = 0$, pelos casos (1), (2) e (3), segue-se que $\delta_{rsj} = 0$.

Caso 5)

Para toda tripla (r,s,j) em $V \times V \times V$, o vetor $\mathcal{X} = (0^{n^2}, 0^n, , 0^{mn}, e^{2mn}_{rsj})$ está em $\mathcal{F}$. Portanto, $(\pi, \beta, \delta, \mu)\mathcal{X} = \pi_0$ e, assim, concluímos que $\mu_{rsj} = 0$.

Até o momento, temos:

- $\pi_0 = 0$ e $\beta_j = 0$ para todo $j \in V$, com $j \neq i$;
- $\alpha_{rj} = 0$ para todo $r, j \in V$, com $(r,j) \neq (u,i)$;
- $\delta_{rsj} = 0$ para todo $r, s, j \in V$, com $(r,s,j) \neq (u,v,i)$ e $r < s$;
- $\mu_{rsj} = 0$ para todo $r, s, j \in V$, com $(r,s,j) \neq (u,v,i)$ e $r \neq s$.

Falta calcularmos os coeficientes de $\alpha_{ui}, \alpha_{vi}, \delta_{uvi}$ e β_i.

Caso 6) $u \in V_i$.

Já que $\mathcal{X}_u^{V_i} \in \mathcal{F}$, tem-se que $\alpha_{ui} + \beta_i = \pi_0$. Sabendo-se que $\pi_0 = 0$, a partir do caso (1), obtemos $\alpha_{ui} = -\beta_i = \theta$. Note que o mesmo argumento pode ser usado ao substituirmos o vértice u pelo vértice v e, assim, temos que $\alpha_{vi} = \theta$.

Caso 7) $u, v \in V_i$.

Como $\mathcal{X}_{u,v}^{V_i} \in \mathcal{F}$, segue-se que $\alpha_{ui} + \alpha_{vi} + \delta_{uvi} + \beta_i = \pi_0$. Sabendo-se que $\alpha_{ui} = -\beta_i$, devido ao caso (6), e que $\pi_0 = 0$, devido ao caso (1), temos que $\alpha_{vi} = -\delta_{uvi} = \theta$.

Assim, concluímos que $\alpha_{ui} = \alpha_{vi} = -\beta_i = -\delta_{uvi} = \theta$.

Após deduzirmos os múltiplos escalares positivos nos casos (1)-(7), temos que a desigualdade $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ é igual a $\theta \times (x_{ui} + x_{vi} - p_{uvi} - y_i \leq 0)$. Com isto finalizamos a prova.

□

3.2 Outras desigualdades válidas para $\mathcal{Q}$

Nesta seção, descreveremos algumas desigualdades válidas que encontramos para o politopo $\mathcal{Q}$ no decorrer dos nossos experimentos computacionais. Estas desigualdades surgiram como resultado dos estudos que fizemos para obtenção de planos-de-corte que eliminassem soluções fracionárias ótimas das relaxações lineares do politopo $\mathcal{Q}$ usadas em nossos experimentos. Sempre que possível, procuramos caracterizar as desigualdades, de modo a incluí-las como membros de classes mais gerais de desigualdades lineares.

Para fins de apresentação, consideramos a solução fracionária $\mathcal{X}^* = (x^*, y^*, p^*, z^*)$, descrita abaixo e foi encontrada para uma instância geométrica com $n = 15$, $m = 41$ e $B = 155Mbs$:

- $x^*_{u2} = x^*_{u(14)} = y^*_2 = y^*_{14} = \frac{1}{2}$ para todo $u \in V$;
- $x^*_{ui} = y^*_i = 0$ para todo $u \in V$ e para todo $i \in V \setminus \{2, 14\}$;

- $p^*_{uvi} = 0$ para todo $u, v \in V$, com $u < v$, e para todo $i \in V$;
- $z^*_{uvi} = 0$ para todo $u, v \in V$, com $u \neq v$, e para todo $i \in V$.

Assumindo que os vértices u, v e r formem um *triângulo* no grafo G, a primeira desigualdade proposta para cortar $\mathcal{X}^*$ é dada por

$$x_{ui} + x_{vi} + x_{ri} - p_{uvi} - p_{uri} - p_{rvi} - y_i \leq 0. \tag{3.2}$$

A desigualdade (3.2) será chamada de desigualdade *clique triangular*. Deve ser notado que o lado esquerdo desta desigualdade, quando calculado para o vetor $\mathcal{X}^*$, é igual a 1 e, portanto, ela é violada por $\mathcal{X}^*$. A seguir, daremos a prova de validade da desigualdade (3.2).

Lema 3.1 *Para $n \geq 3$, caso os vértices u, v e r formam um* triângulo *no grafo G, a desigualdade (3.2) é válida para o politopo $\mathcal{Q}$.*

Prova:

Inicialmente, daremos os argumentos que provam a validade de (3.2). Para tanto, note que se $y_i = 0$, então os três termos em x do lado esquerdo da desigualdade são nulos e, portanto, ela é trivialmente satisfeita. Já quando $y_i = 1$, é fácil ver que o lado esquerdo da desigualdade é maximizado, se uma ou duas variáveis x assumem o valor 1. Mas nestes casos, respectivamente, nenhuma ou apenas uma variável p será obrigada a assumir o valor 1, também. Assim, em ambos os casos, o valor do lado esquerdo da desigualdade será nulo. Logo, a desigualdade (3.2) é válida.

□

O teorema, a seguir, fornece uma condição necessária para que a desigualdade (3.2) defina uma faceta para o politopo $\mathcal{Q}$.

Teorema 3.6 *Para $n \geq 3$, caso os vértices u, v e r formam um* triângulo *no grafo G, a desigualdade (3.2) define uma faceta para o politopo $\mathcal{Q}$.*

Prova:

Seja $\mathcal{F} = \{(x, y, p, z) \in \mathcal{Q} : x_{ui} + x_{vi} + x_{ri} - p_{uvi} - p_{uri} - p_{rvi} - y_i = 0\}$ uma face para o politopo $\mathcal{Q}$. Suponha que exista uma desigualdade $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ válida para $\mathcal{Q}$, tal que cada ponto $(x, y, p, z) \in \mathcal{Q}$ satisfaça $(\alpha, \beta, \delta, \mu)(x, y, p, z) = \pi_0$, ou seja, a face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$ definida por $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ está contida em $\mathcal{F}$.

Sabemos pelo Lema 3.1 que a desigualdade (3.2) é válida. Assim, nosso próximo passo é mostrar que a face $\mathcal{F}$ é uma face própria. Para ver que $\mathcal{Q} \setminus \mathcal{F}$ não é vazio, note que, para todo $j \in \{1, \ldots, n\}$, o vetor $(0^{n^2}, e_j^n, 0^{mn}, 0^{2mn})$ está neste conjunto. Por outro lado, como o vetor nulo está em $\mathcal{F}$, temos que $\mathcal{F}$ é uma face própria de $\mathcal{Q}$.

Os casos abaixo concluem a prova.

Caso 1) Como o vetor nulo está em $\mathcal{F}$, segue-se que $\pi_0 = 0$.

Caso 2) Assuma que o agrupamento V_j, com $j \neq i$, esteja ativo.

Neste caso, teremos $y_j = 1$. Como $\mathcal{X}^{V_j} \in \mathcal{F}$, segue-se que $\beta_j = \pi_0 = 0$.

Caso 3) $k \in V_j$, com $(k, j) \neq (u, i)$.

Sabemos que $x_{kj} = y_j = 1$, logo $\mathcal{X}_k^{V_j} \in \mathcal{F}$. Isto implica em $\alpha_{kj} + \beta_j = \pi_0$. Desde que $\beta_j = 0$, devido aos casos (1) e (2), pode-se concluir que $\alpha_{kj} = 0$.

Caso 4) $k, l \in V_j$, com $(k, l, j) \notin \{(u, v, i), (u, r, i), (v, r, i)\}$.

Neste caso, temos $x_{kj} = x_{lj} = p_{klj} = y_j = 1$, ou seja, $\mathcal{X}_{k,l}^{V_j} \in \mathcal{F}$. Logo, $\alpha_{kj} + \alpha_{lj} + \beta_j + \delta_{klj} = \pi_0$. Como $\alpha_{kj} = \alpha_{lj} = \beta_j = \pi_0 = 0$, pelos casos (1), (2) e (3), segue-se que $\delta_{klj} = 0$.

Caso 5) Para toda tripla (k, l, j) em $V \times V \times V$, o vetor $(0^{n^2}, 0^n, 0^{mn}, e_{klj}^{2mn})$ está em $\mathcal{F}$ e, assim, concluímos que $\mu_{klj} = 0$.

Até o momento, já conseguimos encontrar os seguintes coeficientes:

- $\pi_0 = 0$ e $\beta_j = 0$ para todo $j \in V$, com $j \neq i$;
- $\alpha_{kj} = 0$ para todo $k, j \in V$, com $(k, j) \neq (u, i)$;
- $\delta_{klj} = 0$ para todo $k, l, j \in V$, com $(k, l, j) \neq (u, v, i)$, $(k, l, j) \neq (u, r, i)$, $(k, l, j) \neq (v, r, i)$ e $r < s$;

- $\mu_{klj} = 0$ para todo $k, l, j \in V$, com $(k, l, j) \neq (u, v, i)$, $(k, l, j) \neq (u, r, i)$, $(k, l, j) \neq (v, r, i)$ e $r \neq s$.

Falta calcularmos os coeficientes de $\alpha_{ui}, \alpha_{vi}, \alpha_{ri}, \delta_{uvi}, \delta_{uri}, \delta_{vri}$ e β_i.

Caso 6) $u \in V_i$.

Uma vez que $\mathcal{X}_u^{V_i} \in \mathcal{F}$, tem-se que $\alpha_{ui} + \beta_i = \pi_0$. Sabendo-se que $\pi_0 = 0$, a partir do caso (1), obtemos $\alpha_{ui} = -\beta_i = \theta$. O mesmo argumento pode ser repetido para os vértices v e r, substituindo o vértice u. Logo, temos que $\alpha_{ri} = \alpha_{vi} = \theta$.

Caso 7) $u, v \in V_i$.

Como $\mathcal{X}_{u,v}^{V_i} \in \mathcal{F}$, segue-se que $\alpha_{ui} + \alpha_{vi} + \delta_{uvi} + \beta_i = \pi_0$. Sabendo-se que $\alpha_{ui} = -\beta_i$, devido ao caso (6), e que $\pi_0 = 0$, devido ao caso (1), temos que $\alpha_{vi} = -\delta_{uvi} = \theta$.

O mesmo princípio pode ser aplicado aos vetores de incidência $\mathcal{X}_{u,r}^{V_i} \in \mathcal{F}$ e $\mathcal{X}_{r,v}^{V_i} \in \mathcal{F}$. Assim, podemos concluir que $\alpha_{ui} = \alpha_{vi} = \alpha_{ri} = -\beta_i = -\delta_{uvi} = -\delta_{uri} = -\delta_{vri} = \theta$.

Após deduzirmos os múltiplos escalares positivos nos casos (1)-(7), temos que a desigualdade $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ é igual a $\theta \times (x_{ui} + x_{vi} + x_{ri} - p_{uvi} - p_{uri} - p_{vri} - y_i \leq 0)$. Com isto, conseguimos finalizar a prova.

□

A desigualdade *clique triangular* pode ser generalizada da seguinte forma. Seja $S = (V(S), E(S))$ um subgrafo completo de G. Dado $i \in \{1, \ldots, |V|\}$, considere a notação abaixo:

$$\begin{aligned} x_i(S) &= \sum_{u \in V(S)} x_{ui}, \\ p_i(S) &= \sum_{\substack{u,v \in V(S) \\ u<v}} p_{uvi}. \end{aligned}$$

Lema 3.2 *Seja $S = (V(S), E(S))$ um subgrafo completo de $G = (V, E)$ e $i \in \{1, \ldots, |V|\}$. Se $|V(S)| \geq 2$, α inteiro e $1 \leq \alpha \leq |V(S)| - 2$, a*

desigualdade clique triangular generalizada

$$\alpha x_i(S) - p_i(S) - \frac{\alpha(\alpha+1)}{2} y_i \leq 0 \tag{3.3}$$

é válida para o politopo Ω.

Prova: A validade da desigualdade é trivialmente verificada quando $y_i = 0$. Portanto, suponha que $y_i = 1$. Seja U_i o conjunto de vértices que são alocados ao agrupamento V_i numa solução viável qualquer. Seja $|V(S) \cap U_i| = s$. O lado esquerdo da desigualdade torna-se $\alpha s - s(s-1)/2$.

Se a desigualdade (3.3) é válida, então $\alpha s - s(s-1)/2 - \alpha(\alpha+1)/2$ deve ser menor ou igual a zero, ou ainda, $-\frac{1}{2}[s^2 - (2\alpha+1)s + \alpha(\alpha+1)] \leq 0$. Ocorre que este polinômio em s tem como raízes α e $\alpha+1$, portanto, inteiras. Assim, para qualquer valor inteiro s, o polinômio tem valor positivo, ou seja, o lado esquerdo da desigualdade é negativo. Logo, a desigualdade é válida.

□

As duas próximas desigualdades também cortam a solução fracionária $\mathcal{X}^*$, dada no início desta seção. Para uma aresta (u, v) do grafo G, dado como instância do SRAP, estas desigualdades tomam a forma abaixo:

$$x_{ui} + x_{uj} - z_{uvj} - p_{uvj} - y_i \leq 1, \tag{3.4}$$

$$x_{ui} + x_{uj} + x_{vi} - z_{uvj} - p_{uvi} \leq 1. \tag{3.5}$$

Lema 3.3 *Se* $n \geq 2$, $i \in \{1, \ldots, |V|\}$ *e* (u, v) *é uma aresta de* G, *então a desigualdade (3.4) é válida para o politopo* Ω.

Prova:

Assuma que o vértice u pertence ao agrupamento i. Caso o vértice v pertença ao agrupamento i, teremos p_{uvi} igual a 1. Logo, a desigualdade (3.4) mantém-se válida. Por outro lado, se o vértice v não pertence ao agrupamento i, temos que o lado direito da desigualdade (3.4) será igual a 1. Conseqüentemente, a desigualdade continua válida.

Assuma, agora, que o vértice u não pertence ao agrupamento i. Logo, temos p_{uvi} igual a 0. Entretanto, surge uma questão: o vértice u pertence ou não ao agrupamento j? Caso ele pertença, temos que o lado esquerdo será, no máximo, igual a 1. Por outro lado, se o vértice u não pertence ao agrupamento j, temos que o lado esquerdo da desigualdade (3.4) será, no máximo, igual a 1, devido à variável x_{vi}.

□

Em seguida, enunciamos a prova de validade para a desigualdade (3.5).

Lema 3.4 *Se $n \geq 2$, $i \in \{1, \ldots, |V|\}$ e (u, v) é uma aresta de G, então a desigualdade (3.5) é válida para o politopo $\mathcal{Q}$.*

Prova:

Note que a soma dos dois primeiros termos do lado esquerdo da desigualdade é sempre menor ou igual a 1. Assim, se a desigualdade for violada por alguma solução viável, os três termos em x deverão satisfazer $x_{vi} = 1$ e $x_{ui} + x_{uj} = 1$. Duas situações são possíveis. Se $x_{ui} = 1$, a desigualdade é verificada, pois teríamos que $p_{uvi} = 1$. Por outro lado, se $x_{uj} = 1$, a desigualdade é verificada, pois teríamos que $z_{uvj} = 1$.

□

O teorema enunciado, a seguir, estabelece uma condição necessária para que a desigualdade (3.5) defina uma faceta para o politopo $\mathcal{Q}$.

Teorema 3.7 *Para $n \geq 2$, se $i \in \{1, \ldots, |V|\}$ e (u, v) é uma aresta de G, então a desigualdade (3.5) define uma faceta para o politopo $\mathcal{Q}$.*

Prova:

Seja $\mathcal{F} = \{(x, y, p, z) \in \mathcal{Q} : x_{ui} + x_{uj} + x_{vi} - z_{uvj} - p_{uvi} = 1\}$ uma face para o politopo $\mathcal{Q}$. Suponha que exista uma desigualdade $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ válida para $\mathcal{Q}$, tal que cada ponto $(x, y, p, z) \in \mathcal{Q}$ satisfaça $(\alpha, \beta, \delta, \mu)(x, y, p, z) = \pi_0$, ou seja, a face $\mathcal{F}_{(\alpha,\beta,\delta,\mu)}$, definida por $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$, está contida em $\mathcal{F}$.

Sabemos pelo Lema 3.4 que a desigualdade (3.5) é válida. Assim, nosso próximo passo é mostrar que a face $\mathcal{F}$ é uma face própria. Note que o vetor nulo pertence a $\mathcal{Q} \setminus \mathcal{F}$, e que o vetor $(e_{ui}^{n^2}, e_i^n, 0^{mn}, 0^{2mn})$ está em $\mathcal{F}$. Deste modo, $\mathcal{F}$ é uma desigualdade própria de $\mathcal{Q}$.

Os casos enumerados abaixo concluem a demonstração desta prova.

Caso 1) $u \in V_j$, com $j \neq i$.

Neste caso, temos x_{uj} e $y_j = 1$. Como $\mathcal{X}_u^{V_j} \in \mathcal{F}$, segue-se $\alpha_{uj} + \beta_j = \pi_0$.

Caso 2) Considere que o agrupamento V_k esteja ativo, com $k \neq j$.

Neste caso, temos $y_k = 1$. Como $\mathcal{X}^{V_k} \in \mathcal{F}$, temos que $\alpha_{uj} + \beta_j + \beta_k = \pi_0$. Sabemos que $\alpha_{uj} + \beta_j = \pi_0$, devido ao caso (1). Logo, concluímos que $\beta_k = 0$.

Caso 3) $u \in V_i$, com $i \neq j$.

Sabemos que $x_{ui} = y_i = 1$. Através do vetor de incidência $\mathcal{X}_u^{V_i} \in \mathcal{F}$, obtemos $\alpha_{ui} + \beta_i = \pi_0$. Como $\beta_i = 0$, devido ao caso (2), pode-se concluir que $\alpha_{ui} = \pi_0$.

Caso 4) $u \in V_i$, com $i \neq j$. Considere, ainda, que o agrupamento V_j esteja ativo.

Neste caso, temos $x_{ui} = y_i = y_j = 1$. Conseqüentemente, $\mathcal{X}_u^{V_i,V_j} \in \mathcal{F}$. Assim, obtemos $\alpha_{ui} + \beta_i + \beta_j = \pi_0$. Pelo caso (2), conclui-se que $\beta_j = 0$. Através do caso (1), conclui-se ainda que $\alpha_{uj} = \pi_0$.

Caso 5) $v \in V_i$, com $i \neq j$.

Sabemos que $x_{vi} = y_i = 1$. Através do vetor de incidência $\mathcal{X}_v^{V_i} \in \mathcal{F}$, obtemos $\alpha_{vi} + \beta_i = \pi_0$. Como $\beta_i = 0$, devido ao caso (2), pode-se concluir que $\alpha_{vi} = \pi_0$.

Caso 6) $u, v \in V_i$, com $i \neq j$.

Neste caso, temos $x_{ui} = x_{vi} = p_{uvi} = y_i = 1$. Isto implica que $\mathcal{X}_{u,v}^{V_i} \in \mathcal{F}$. Logo, obtemos $\alpha_{ui} + \alpha_{vi} + \delta_{uvi} + \beta_i = \pi_0$. Sabendo-se que $\alpha_{ui} = \alpha_{vi} = \pi_0$, devido aos casos (4) e (5), e que $\beta_i = 0$, devido ao caso (2), conclui-se que $\delta_{uvi} = -\pi_0$.

Caso 7) $u \in V_j$ e $v \in V_i$.

Neste caso, temos $x_{uj} = x_{vi} = z_{uvj} = y_i = y_j = 1$, ou seja, $\mathcal{X}_{v,u}^{V_i,V_j} \in \mathcal{F}$. Logo, $\alpha_{uj} + \alpha_{vi} + \mu_{uvj} + \beta_i + \beta_j = \pi_0$. Como $\alpha_{ui} = \alpha_{vi} = \pi_0$, devido aos casos (4) e (5), e $\beta_i = 0$, devido ao caso (2), conclui-se que $\delta_{uvi} = -\pi_0$.

Caso 8) Considere toda tripla (r, s, k) em $V \times V \times V$, e $u \in V_i$, com $(r, s, k) \neq (u, v, i)$. O vetor $\mathcal{X}$, com todas as componentes nulas, excetuando-se as componentes p_{rsk}, x_{ui} e y_i, cujos valores são iguais a 1, está em $\mathcal{F}$. Com isto, temos $\alpha_{ui} + \beta_i + \delta_{rsk} = \pi_0$. Pelos casos (2) e (3), sabemos que $\beta_i = 0$ e $\alpha_{ui} = 1$, respectivamente. Logo, temos $\delta_{rsk} = 0$ para todas as arestas do grafo e todos os agrupamentos.

Caso 9) Analogamente, considere toda tripla (r, s, k) em $V \times V \times V$, e $u \in V_j$, com $(r, s, k) \neq (u, v, j)$. Isto implica que o vetor $\mathcal{X}$, com todas as componentes nulas, excetuando-se as componentes z_{rsk}, x_{uj} e y_j, cujos valores são iguais a 1, está em $\mathcal{F}$. Assim, temos $\alpha_{uj} + \mu_{rsk} + \beta_j = \pi_0$. Através do caso (4), conclui-se que $\mu_{rsj} = 0$ para toda aresta do grafo e todo agrupamento, à exceção da aresta (u, v) quando o agrupamento for i.

□

Assim, concluímos a prova uma vez que $\alpha x + \beta y + \delta p + \mu z \leq \pi_0$ pode ser escrita a partir da desigualdade (3.5) e dos múltiplos escalares positivos encontrados nos casos (1)-(9).

3.3 Resultados computacionais

Nesta seção, apresentamos os resultados computacionais obtidos com o procedimento *branch-and-bound*, baseado na formulação (P2) e em algumas estratégias constituídas por esta formulação, adicionadas de algumas desigualdades discutidas nas seções anteriores. O enfoque desta seção é a melhoria da formulação (P2).

As instâncias utilizadas são as mesmas descritas no capítulo anterior. Todos os experimentos foram realizados em um computador com processador `AMD K6 450MHz` e com `256Mbytes` de memória `RAM`. Para a execução do procedimento *branch-and-bound*, utilizou-se o resolvedor linear inteiro do pacote

de otimização XPRESS 12.05 (vide [14]).

3.3.1 Estratégias

A escolha de uma formulação inteira que proporcionasse bons limitantes inferiores foi uma das primeiras preocupações que tivemos na resolução, de forma exata, do problema de atribuição de localidade a anéis. No Capítulo 2 foram apresentadas duas formulações inteiras para o SRAP. A formulação (P2), que propusemos, foi a que apresentou um melhor desempenho computacional, segundo alguns critérios.

Nesta seção, procuramos verificar se com a inclusão das desigualdades propostas, que são supostamente mais fortes, a formulação (P2) nos trará alguma melhoria no desempenho computacional desta formulação.

Como visto anteriormente, foram definidas algumas famílias de facetas para o politopo Q, associado ao SRAP. Em seguida, apresentamos algumas estratégias, para definir qual daquelas desigualdades podem ser adicionadas à formulação (P2).

Estratégia F1: *o modelo é formado por todas as desigualdades (3.1) mais as desigualdades da formulação (P2).*

Estratégia F2: *o modelo é formado por todas as desigualdades (3.2) mais as desigualdades da formulação (P2).*

As Tabelas 3.1 e 3.2 mostram os resultados obtidos com o emprego das estratégias F1 e F2, respectivamente. A primeira coluna indica a instância testada; a segunda, o ótimo inteiro; a terceira coluna indica o número de nós abertos na árvore *branch-and-bound*; a quarta, o limitante inferior obtido e, por último, a quinta coluna indica o tempo computacional, expresso em segundos, gasto na resolução da instância.

Nenhuma das estratégias conseguiu melhorar o valor da relaxação linear em relação à formulação (P2) original. Assim, como este seria um critério natural para avaliar a qualidade das facetas, nossa avaliação será fundamentada em outros critérios, por exemplo, o desempenho computacional.

Claramente, observa-se através destes experimentos que a estratégia F1 é

bem mais eficiente do que a estratégia F2. Por exemplo, o tempo obtido para a instância gl_15_9 foi quase cinco vezes menor do que o tempo que tínhamos antes. O mesmo desempenho pode ser visto para todas as instâncias testadas, à exceção da instância gh_25_2, onde o tempo computacional aumentou.

Um outro comentário pode ser feito com relação aos experimentos computacionais realizados nesta seção. Ao removermos as desigualdades (2.10) da estratégia F1, visto que estas são dominadas pelas desigualdades (3.1), o resolvedor linear inteiro teve um comportamento inesperado: consumiu um maior tempo computacional. Por exemplo, para a instância gl_15_9, o tempo computacional obtido com a estratégia F1 foi de 381 segundos, enquanto que a versão da estratégia F1, sem as desigualdades (2.10), obteve um tempo computacional de 614 segundos.

		Estratégia F1		
Nome	z^*	# Nds	z	Tmp
gl_15_1	3	2935	3	585
gl_15_4	3	2078	3	397
gl_15_9	3	6080	3	381
gh_15_1	3	4025	3	517
gh_15_2	3	2292	3	465
gh_15_9	3	4019	3	479
gh_25_2	3	3170	3	1731
rl_15_1	3	988	3	97
rl_25_8	3	1274	3	510
rh_15_1	3	1645	3	192
rh_15_5	3	4156	3	424

Tabela 3.1: Desempenho do procedimento *branch-and-bound* para instâncias geométricas e aleatórias da classe C1, segundo a estratégia F1.

Apesar desta melhoria, ainda que aparente, na aplicação das desigualdades (3.1), podemos enumerar duas dificuldades que ainda permanecem com a resolução de forma exata do SRAP. A primeira refere-se ao valor obtido na relaxação linear no primeiro nó da árvore *branch-and-bound*, o qual, como dito anteriormente, ainda continua sendo igual a 1, qualquer que seja a estratégia utilizada. Estes limitantes duais fracos, obtidos pelas relaxações lineares, resultam em uma segunda dificuldade, que é o aumento do número de nós explorados pelo procedimento *branch-and-bound*.

		Estratégia F2		
Nome	z^*	# Nds	z	Tmp
gl_15_1	3	3116	3	2021
gl_15_4	3	1904	3	2100
gl_15_9	3	9510	3	3759
gh_15_1	3	1806	3	1140
gh_15_2	3	6528	3	3860
gh_15_9	3	4253	3	5007
gh_25_2	3	5830	3	13802
rl_15_1	3	5500	3	1752
rl_25_8	3	1281	3	1966
rh_15_1	3	3705	3	1995
rh_15_5	3	1841	3	1385

Tabela 3.2: Desempenho do procedimento *branch-and-bound* para instâncias geométricas e aleatórias da classe C1, segundo a estratégia F2.

Os experimentos realizados com algumas instâncias do Capítulo 2, e reproduzidos na Tabela 3.3, ilustram as dificuldades encontradas ao tentarmos resolver instâncias de médio porte com a estratégia F1. À exceção da instância rh_25_2, que foi resolvida à otimalidade dentro do limite de tempo computacional estabelecido, em todas as demais instâncias, verificou-se a exploração de um elevado número de nós na árvore de enumeração e a interrupção da execução do procedimento *branch-and-bound* pela limitação de tempo.

Na próxima seção, apresentamos maneiras de reformular o problema de atribuição de localidades a anéis, cujo intuito é superar as dificuldades expostas acima.

3.4 Reduzindo a simetria das soluções

Como apresentado anteriormente, o problema de atribuição de localidades a anéis pode ser modelado como um problema de programação linear inteira 0-1. Entretanto, existe uma dificuldade inerente à formulação do SRAP: a simetria entre as soluções.

A simetria no problema de atribuição de localidades a anéis ocorre, porque um mesmo conjunto de agrupamentos viáveis pode ter várias representações

	Formulação (P2)		Estratégia F1	
Nome	# Nds	z	# Nds	z
gl_25_1	13500	2	13200	2
gl_25_3	13500	2	11600	2
gl_25_4	9400	2	14900	2
gl_25_8	16100	2	13000	2
gh_25_7	4300	2	22100	2
rl_25_4	9200	2	14000	2
rl_25_5	9100	2	20000	2
rl_25_6	8200	2	11600	2
rl_25_9	23100	2	16200	2
rl_25_10	7600	2	14700	2
rh_25_2	15600	2	4050	3

Tabela 3.3: Desempenho do procedimento *branch-and-bound* para instâncias da classe C1, segundo a estratégia F1.

no modelo matemático. Estudos realizados sobre simetria entre as soluções para outros problemas combinatórios foram feitos por Holm e Sorensen [29] e, Sherali e Smith [66].

As Figuras 3.1 e 3.2 mostra um exemplo da simetria entre soluções para um grafo com $n = 25$ e $B = 155Mbs$. Nas duas figuras, temos que o número mínimo de anéis é igual a 3, porém a atribuição das localidades aos anéis é diferente.

Na verdade, observamos que, mesmo para instâncias pequenas, essa dificuldade já é refletida na resolução do problema de programação linear inteira 0-1. Isso faz com que o processo de enumeração no procedimento *branch-and-bound* consuma muito tempo, pois existe a necessidade de enumerar muitos nós na árvore *branch-and-bound*. Além disso, esta simetria dá origem a uma diferença percentual muito grande entre o valor da relaxação linear e o ótimo inteiro.

3.4.1 Desigualdades lineares

Diante das dificuldades expostas anteriormente, dedicamo-nos ao estudo de como poderíamos quebrar a simetria das soluções do problema de atribuição de localidades a anéis, através da inclusão de desigualdades lineares à formulação (P2).

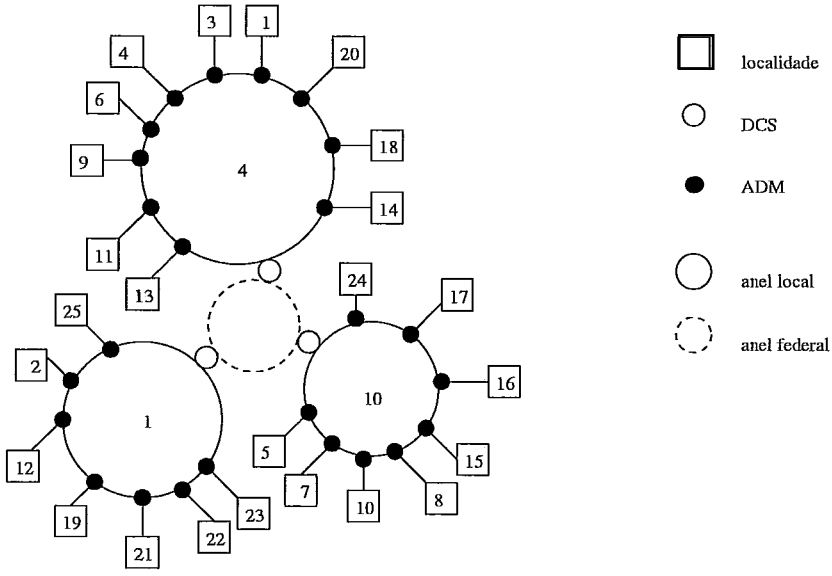

Figura 3.1: Solução S1 obtida para uma instância com $n = 15$ e $B = 155Mbs$.

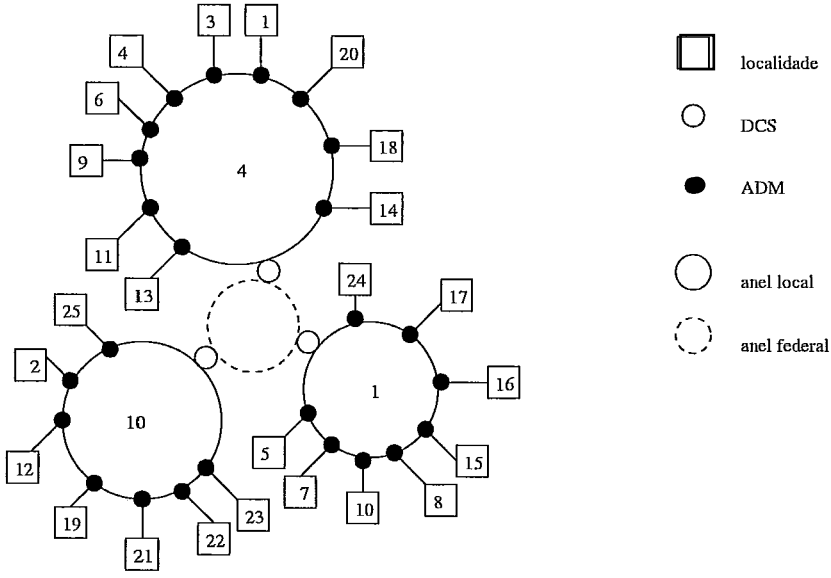

Figura 3.2: Solução simétrica à solução S1.

Idealmente, o novo modelo deverá ser capaz de reduzir a diferença entre o valor da relaxação linear e o ótimo inteiro, e a quantidade de nós enumerados na árvore *branch-and-bound*. As desigualdades que experimentamos para atingir estes fins são descritas a seguir.

Desigualdade 1: estabelece uma relação de hierarquia entre os agrupamentos, impondo que agrupamentos de menor índice devem ser usados com uma prioridade maior que os agrupamentos de maior índice.

$$y_i - y_{i-1} \leq 0, \qquad i = 2, \ldots, n. \tag{3.6}$$

Desigualdade 2: estabelece um outro tipo de hierarquia entre os agrupamentos, impondo que para os agrupamentos de menor índice não devem ser alocados menos vértices do que àqueles de maior índice.

$$\sum_{u=1}^{n} x_{ui} - \sum_{u=1}^{n} x_{u(i-1)} \leq 0, \qquad i = 2, \ldots, n. \tag{3.7}$$

Em seguida, apresentamos alguns resultados computacionais obtidos com o emprego destas desigualdades, na resolução do problema de atribuição de localidades a anéis.

3.4.2 Resultados computacionais

Os resultados que obtivemos até o momento são para instâncias consideradas pequenas ($n = 15$ ou $n = 25$), e com o emprego das desigualdades (3.1) adicionadas à formulação (P2).

Nesta subseção, definiremos algumas estratégias, com o intuito de verificarmos o quanto as desigualdades apresentadas na seção 3.4.1 são eficientes na resolução de instâncias do problema de atribuição de localidades a anéis. As estratégias são:

Estratégia S1: *geramos todas as desigualdades (3.6) e as adicionamos à formulação (P2).*

Estratégia S2: *geramos todas as desigualdades (3.7) e as adicionamos à formulação (P2).*

Os critérios empregados desta vez, para determinarmos a escolha da melhor estratégia, foram o valor obtido no primeiro nó da árvore *branch-and-bound*, o número de nós abertos e o tempo computacional envolvido.

Experimentos

As Tabelas 3.4 e 3.5 apresentam, respectivamente, um quadro comparativo entre as estratégias S1 e S2. Cada tabela está dividida em duas partes. A primeira parte consiste nos resultados obtidos com a formulação (P2). Isso corresponde à segunda e terceira colunas. Na segunda coluna, indicamos o número de planos-de-cortes do pacote de otimização XPRESS que foram adicionados à relaxação linear. A terceira coluna indica o valor obtido na relaxação linear.

A segunda parte, daquelas tabelas, consiste na apresentação dos resultados obtidos com a formulação (P2), acrescida de todas as desigualdades definidas em cada estratégia. Isso corresponde à quarta, quinta, sexta, sétima e oitava colunas. O significado da quarta e quinta colunas é o mesmo da segunda e terceira colunas, respectivamente. A sexta coluna indica o número de nós abertos, a sétima coluna, o limitante obtido, e a oitava coluna, indica o tempo computacional gasto.

Como se pode ver, os resultados obtidos com o emprego de desigualdades lineares, para reduzir a simetria das soluções, melhoraram sensivelmente com relação aos resultados apresentados no Capítulo 2 e na seção anterior.

Um exemplo desta melhoria pode ser verificada quando observarmos a Estratégia S1. Para a instância `gl_15_1`, o tempo computacional é quase 10 vezes menor do que o obtido pela formulação (P2). Esta melhoria também ocorre quando comparamos o resultado obtido pela Estratégia F1 para esta instância. Neste caso, o desempenho da Estratégia S1 foi quase 9 vezes melhor do que o da Estratégia F1.

Além disso, a estratégia conseguiu melhorar, pela primeira vez, o valor da relaxação linear no primeiro nó da árvore *branch-and-bound*. Este ganho

para as instâncias gl_15_1, gh_15_9, rl_15_1 e rh_15_1, por exemplo, chega algumas vezes a ultrapassar 50%.

	Formulação (P2)		Estratégia S1				
Nome	# Crts	$\overline{z}$	# Crts	$\overline{z}$	# Nds	z	Tmp
gl_15_1	7	1	18	1.60	1511	3	61
gl_15_4	19	1	32	1.45	1470	3	44
gl_15_9	3	1	18	1.36	441	3	13
gh_15_1	2	1	126	1.30	1128	3	131
gh_15_2	15	1	23	1.28	344	3	31
gh_15_9	2	1	17	2	888	3	51
gh_25_2	2	1	52	1.21	667	2	106
rl_15_1	9	1	61	1.86	1086	3	14
rl_25_8	2	1	32	1.25	3212	3	98
rh_15_1	10	1	6	2	345	3	28
rh_15_5	2	1	42	1.44	548	3	46

Tabela 3.4: Redução da simetria para instâncias geométricas e aleatórias da classe C1, segundo a estratégia S1.

	Formulação (P2)		Estratégia S2				
Nome	# Crts	$\overline{z}$	# Crts	$\overline{z}$	# Nds	z	Tmp
gl_15_1	7	1	4	1	7359	3	300
gl_15_4	19	1	4	1	898	3	20
gl_15_9	3	1	8	1	2206	3	38
gh_15_1	2	1	15	1	2280	3	103
gh_15_2	15	1	30	1	1556	3	164
gh_15_9	2	1	14	1	3063	3	184
gh_25_2	2	1	13	1	3853	2	908
rl_15_1	9	1	16	1	4400	3	94
rl_25_8	2	1	4	1	63993	3	11207
rh_15_1	10	1	18	1	3080	3	277
rh_15_5	2	1	3	1	2169	3	164

Tabela 3.5: Redução da simetria para instâncias geométricas e aleatórias da classe C1, segundo a estratégia S2.

Pode-se observar que com a estratégia S1 chega-se a resultados melhores do que com a estratégia S2. À primeira vista, isto pode parecer surpreendente pois, para soluções inteiras, a estratégia S2 é mais restritiva do que a estratégia S1.

Considere que para uma dada instância haja uma única solução ótima e que esta solução contenha cinco agrupamentos. Além disso, suponha que o

número de vértices em cada agrupamento é distinto. A estratégia S1 ainda teria pelo menos 120 vetores inteiros ótimos que representariam a mesma solução física (obtidos através de permutação dos índices referentes aos agrupamentos de 1 a 5). Já com a estratégia S2, haveria apenas um único vetor de incidência associado a esta solução (o agrupamento com mais vértices seria associado ao índice 1, o agrupamento com o segundo maior número de vértices seria associado ao índice 2 e assim por diante). Aparentemente, a razão para isso se deve à maior capacidade do XPRESS de encontrar planos-de-corte que melhoram o modelo referente à estratégia S1. Este fato pode ser notado, comparando-se a quarta coluna das Tabelas 3.4 e 3.5. Esta razão, provavelmente, explica também porque com a estratégia S1 foi possível, pela primeira vez, melhorar o valor da relaxação linear no primeiro nó da árvore *branch-and-bound*.

Devido ao seu melhor desempenho, empregamos a estratégia S1 também na resolução de instâncias do SRAP com $n = 25$. Todas essas instâncias, anteriormente, tinham um tempo computacional igual a 18,000 segundos (tempo máximo). A Tabela 3.6 apresenta esses resultados. Esta tabela está dividida em duas partes iguais. A primeira parte consiste nos resultados obtidos com a formulação (P2), enquanto que a segunda parte consiste nos resultados obtidos com a Estratégia S1. A coluna `# Crts` indica o número de planos-de-cortes do pacote XPRESS que foram adicionados à relaxação linear. A coluna $\overline{z}$ indica o valor da relaxação linear. A coluna `# Nds` indica o número de nós abertos na árvore *branch-and-bound*. E, por último, a coluna z indica o limitante inferior obtido no tempo máximo.

Novamente, apenas para a instância `rh_25_2` foi possível provar a otimalidade antes de 18,000 segundos. Note que para esta instância exploramos menos nós na árvore *branch-and-bound*, e isto pode ser um indicativo de que a estratégia S1 seja melhor do que a estratégia F1. Além disso, nas demais instâncias, conseguimos melhorar o valor do limitante inferior z, os valores agora são maiores do que 2. Se compararmos estes valores com os valores obtidos com a estratégia F1, podemos deduzir que tais instâncias não possuem soluções ótimas com apenas dois agrupamentos.

	Formulação (P2)				Estratégia S1			
Nome	# Crts	$\overline{z}$	# Nds	z	# Crts	$\overline{z}$	# Nds	z
gl_25_1	5	1	13500	2	62	1.31	10400	2.33
gl_25_3	2	1	13500	2	27	1.28	10600	2.23
gl_25_4	2	1	9400	2	70	1.32	10700	2.33
gl_25_8	2	1	16100	2	13	1.37	900	2.04
gh_25_7	2	1	4300	2	26	1.30	5000	2.03
rl_25_4	2	1	9200	2	25	1.28	22400	2.26
rl_25_5	15	1	9100	2	27	1.28	36200	2.32
rl_25_6	2	1	8200	2	25	1.29	18200	2.37
rl_25_9	2	1	23100	2	8	1.29	32700	2.31
rl_25_10	20	1	7600	2	40	1.44	12500	2.25
rh_25_2	2	1	15600	2	37	1.21	435	3

Tabela 3.6: Resultados computacionais com a Estratégia S1 em instâncias geométricas e aleatórias da classe C1, com $n = 25$.

3.5 Considerações

Apesar do sucesso obtido na resolução de algumas instâncias de pequeno e médio porte do problema de atribuição de localidades a anéis, com o emprego das estratégias F1 e S1, ainda nos deparamos com dois problemas: melhorar o valor da relaxação linear e procurar diminuir o número de nós na árvore de enumeração. Estes fatores, quando solucionados, provavelmente resultarão em um melhor desempenho computacional.

Essa dificuldade na resolução, de forma exata, das instâncias do problema de atribuição de localidade a anéis deve-se, ainda, à questão da simetria das soluções. No capítulo 5, procuramos superar este obstáculo, através de uma reformulação do SRAP e do uso da técnica de geração de colunas.

Capítulo 4

Limitantes superiores

Em um algoritmo *branch-and-bound*, o cálculo dos limitantes inferiores e superiores é muito importante e merece uma atenção especial. Para um problema de minimização, o valor da função objetivo, obtido a partir da resolução do problema de problema de programação linear em cada nó, corresponde ao limitante inferior. Já o limitante superior pode ser obtido através de duas maneiras distintas: a primeira, na fase de inicialização do algoritmo *branch-and-bound*; a segunda, durante a execução deste, através do uso da informação contida nas variáveis correspondentes à solução ótima das relaxações lineares.

Neste capítulo, apresentaremos uma metaheurística utilizada na fase de inicialização de um algoritmo *branch-and-bound* para a obtenção de limitantes superiores (iniciais) para o problema de atribuição de localidades a anéis.

Antes de iniciarmos a discussão sobre a metaheurística que propomos, vale ressaltar que o objetivo aqui é apenas o de fornecer um limitante primal inicial para um algoritmo *branch-and-bound*. Portanto, ainda que isto seja factível, não nos empenhamos em fazer testes exaustivos de ajustes de parâmetros com o intuito de superar as heurísticas existentes na literatura.

4.1 Trabalhos anteriores

Nesta seção, apresentaremos dois trabalhos propostos para a resolução, de forma heurística, do problema de atribuição de localidades a anéis. O primeiro trabalho faz uso de três heurísticas gulosas, enquanto que o segundo utiliza uma metaheurística busca tabu. Em seguida, apresentamos uma breve

descrição destes trabalhos.

4.1.1 Heurísticas gulosas

Uma heurística é um algoritmo projetado para encontrar uma boa solução, mas para a qual não se pode dar uma garantia da qualidade em relação à solução ótima. As heurísticas são empregadas para encontrar soluções razoáveis para problemas pertencentes à classe NP-difícil.

Uma heurística é única, ou seja, só pode ser aplicada àquele problema específico para o qual ela foi projetada. Porém, qualquer heurística possui dois aspectos em comum: rapidez e alguma estratégia aplicada ao seu projeto.

Quando nos referimos à rapidez, se exige da heurística que ela consiga produzir uma solução "satisfatória" em uma quantidade de tempo razoável, mesmo quando a entrada do problema aumenta exponencialmente.

Já as estratégias se referem a possíveis maneiras de como podemos gerar uma solução. Duas são as estratégias comumente utilizadas: estratégia gulosa e busca local. As heurísticas que descreveremos nesta seção fazem uso da estratégia gulosa.

A estratégia gulosa é uma das mais comuns, e talvez a mais importante das estratégias utilizadas no projeto de heurísticas. A solução é construída passo-a-passo, e em cada um desses passos um elemento é selecionado para ser adicionado à solução.

Em Goldschmidt *et al.* [26], os autores propuseram três heurísticas gulosas para resolver o problema de atribuição de localidades a anéis: *edge-based*, *cut-based* e *node-based*. Uma pequena descrição dessas heurísticas é apresentada logo abaixo.

Edge-based e *Cut-based*

As heurísticas *edge-based* e *cut-based*, inicialmente, atribuem cada localidade a um anel diferente; em seguida, a cada iteração, dois anéis distintos são unidos, caso o anel resultante desta união seja viável. Na heurística *edge-based*, as arestas são ordenadas em ordem crescente de suas demandas, e a escolha dos anéis para a união é feita, tomando-se a aresta de maior peso,

que ainda não foi selecionada. Na heurística *cut-based*, a escolha dos anéis é feita, tomando-se o tráfego máximo entre dois anéis quaisquer.

Node-based

Já a heurística *node-based* recebe um valor k, correspondente ao número inicial de anéis. Estes anéis estão vazios *a priori*. Em seguida, escolhe-se, de forma aleatória, uma localidade que será atribuída a um anel vazio. Após a atribuição destas localidades, começa-se a trabalhar com as localidades que ainda não foram atribuídas a um anel. Neste caso, a cada iteração escolhe-se um anel i que possua a menor capacidade e, em seguida, escolhe-se uma localidade u que apresente o maior trafégo com o anel i.

4.1.2 Metaheurísticas

Uma metaheurística consiste em várias heurísticas de caráter genérico que se adaptam facilmente às estruturas de arquiteturas paralelas e são direcionadas a uma otimização global de um problema, podendo conter diferentes procedimentos heurísticos de busca local em sua estrutura.

Nas últimas décadas, surgiram vários procedimentos, enquadrados como metaheurísticas, empregados na resolução de problemas de otimização combinatória. Dentre elas, destacam-se: algoritmos genéticos, busca tabu, *simulated annealing* e, mais recentemente, *A-Teams*, *ant colony*, *variable neighborhood search* e *GRASP*.

Em seguida, apresentamos uma pequena descrição de uma metaheurística busca tabu, empregada na resolução do SRAP.

Busca tabu

Em Aringhieri e Dell'Amico [3], os autores propuseram uma busca tabu para a resolução do problema de atribuição de localidades a anéis. Para essa metaheurística, foram implementadas duas estratégias de vizinhança.

A primeira vizinhança, denotada pelos autores por N1, inicia a partir de uma solução (viável ou inviável) e move uma localidade de um anel para outro anel, tal que o anel resultante seja viável.

Já a segunda vizinhança, denotada por N2, trabalha com uma solução obtida a partir da vizinhança N1, com o intuito de diminuir o número de anéis existentes na solução. Para isso, escolhe-se um anel que possua poucas localidades, e as localidades desse anel são distribuídas para os outros anéis, independentemente da viabilidade da solução. Após esvaziar esse anel, a vizinhança N1 é executada novamente.

Além dessas duas estratégias de vizinhança, os autores também implementaram outras estratégias para a busca tabu, com o intuito de intensificar a busca dentro de regiões promissoras ou de diversificá-la, gerando novas soluções. Dentre as estratégias implementadas, temos *path relinking* e *eXploring tabu search.*

4.2 Metaheurística *GRASP*

Nesta seção, descrevemos a metaheurística *GRASP (Greedy Randomized Adaptive Search Procedure)* que propusemos para o problema de atribuição de localidades a anéis. A escolha desta metaheurística deve-se à simplicidade da implementação e de estarmos utilizando este método apenas para a obtenção de limitantes superiores iniciais.

Um *GRASP* é um procedimento iterativo, que combina várias propriedades favoráveis de outras heurísticas (vide Feo e Resende [19]). Mais especificamente, cada iteração do *GRASP* consiste em dois estágios: uma fase de construção da solução e uma fase de busca local. Em cada iteração, uma solução é encontrada. A melhor solução obtida dentre todas as iterações é considerada a solução final.

Em seguida, apresentamos um algoritmo informal para a metaheurística *GRASP*. No passo 1, temos a leitura dos dados de entrada. Nos passos 2-6, temos o *GRASP* propriamente dito. No passo 3, ocorre a fase de construção da solução. No passo 4, é realizada a fase de busca local. Por último, no passo 5, temos a atualização da solução, caso algum melhoramento tenha sido obtido. O critério de parada comumente utilizado é estipular um número máximo de iterações.

Na fase de construção de uma solução, iniciamos com um conjunto vazio,

Algoritmo 1 Algoritmo informal para a metaheurística *GRASP*

1: Leitura dos dados
2: **Enquanto** critério de parada não for satisfeito **faça**
3: Construa uma solução aleatória baseada em um critério guloso
4: Realize uma busca local
5: Atualize a solução
6: **Fim-Enquanto**

que iterativamente recebe um elemento, até formar uma solução viável. Em cada iteração da fase de construção, dois aspectos são analisados: a aleatoriedade e a adaptação. A aleatoriedade se deve ao fato de que o próximo elemento a ser escolhido para compor a solução não será necessariamente o melhor, segundo o critério guloso utilizado. Na verdade, a escolha é feita de forma aleatória, a partir de uma lista, denominada *Lista Restrita de Candidatos* (LRC), que contém os β melhores elementos candidatos segundo o critério guloso especificado. O valor de β é um parâmetro do algoritmo. Por outro lado, a adaptatividade do processo se mostra evidente, à medida que se escolhe um elemento que irá compor a solução inicial, pois os próximos elementos a serem inseridos em LRC dependerão dos já incluídos na solução inicial.

As soluções obtidas na fase de construção do *GRASP* não são garantidas como ótimos locais, considerando uma dada vizinhança (diz-se que uma solução s é *localmente ótima*, se não existe uma solução melhor na vizinhança de s). Portanto, o emprego da segunda fase do *GRASP* é feito com o intuito de se melhorar a solução obtida na fase de construção. Um algoritmo de busca local é utilizado para, sucessivamente, substituir a solução atual por uma solução melhor, encontrada na vizinhança da solução atual. O algoritmo termina quando nenhuma solução melhor é encontrada na vizinhança da solução atual ou após algum outro critério de parada ter sido atingido.

Antes de descrevermos os detalhes de cada uma das fases da metaheurística *GRASP* implementada para o problema de atribuição de localidades a anéis, introduziremos uma notação que facilitará a compreensão daquelas fases.

Seja r o índice de um anel qualquer em uma solução S. Os valores asso-

ciados aos tráfegos deste anel r são:

$$I_r = \sum_{\substack{u,v \in r \\ u<v}} d_{uv}, \tag{4.1}$$

$$E_r = \sum_{\substack{u \in r \\ v \notin r}} d_{uv}, \tag{4.2}$$

$$T_r = I_r + E_r, \tag{4.3}$$

onde I_r, E_r e T_r representam, respectivamente, os tráfegos interno, externo e total no anel r. Note que o anel r é viável se $T_r \leq B$.

Então, com respeito a uma dada solução S, podemos definir

$$F = \frac{\sum_r E_r}{2}, \tag{4.4}$$

$$BFed = F - B, \tag{4.5}$$

$$BNet = \max_r \{T_r - B\}, \tag{4.6}$$

onde F, $BFed$ e $BNet$ indicam, respectivamente, o tráfego total no anel federal, a quantidade de tráfego em excesso no anel federal e a quantidade máxima de tráfego em excesso nos anéis da solução S.

Com o intuito de direcionarmos a busca através de soluções viáveis e com uma boa qualidade, definimos a seguinte função objetivo, em função do número de anéis k e dos valores de $BFed$ e de $BNet$:

$$\min z = k + \alpha BNet + \delta BFed, \tag{4.7}$$

onde α e δ são valores reais positivos.

Essa função z permite que tenhamos soluções com um pequeno número de anéis, ao mesmo tempo que procuramos penalizar soluções inviáveis através

dos parâmetros α e δ, ou seja, tentamos minimizar os valores de $BNet$ e $BFed$.

Definidos esses valores, iniciaremos agora a descrição das fases do *GRASP*.

4.2.1 Fase de construção

Na fase de construção, iniciamos com uma solução S constituída por n anéis, ou seja, cada localidade $u \in V$ é atribuída a um anel de mesmo índice. Além disso, consideramos que cada aresta $(u, v) \in E$ é uma aresta candidata para induzir a união viável entre os dois anéis r e t, quaisquer com $u \in r$ e $v \in t$.

Um outro aspecto que deve ser levantado nesta fase é a obtenção de LRC. A cada iteração, são escolhidas as β melhores arestas (u, v) candidatas de acordo com a demanda d_{uv}. Este valor β, denominado *restrição de cardinalidade*, limita o tamanho de LRC. A lista LRC é obtida em duas etapas. Na primeira, construímos um conjunto de arestas que ainda não foram selecionadas, ou seja, as arestas cujos extremos estão em anéis diferentes. Na segunda etapa, utilizamos o algoritmo Estatística de Ordem (vide Cormen *et al.* [11]), para obter as β primeiras arestas daquele conjunto, sem precisar ordená-las a cada iteração, o que demandaria um maior tempo computacional.

Por último, após a obtenção de LRC, selecionamos uma aresta $(u, v) \in LRC$ de forma aleatória, onde $u \in r$ e $v \in t$. Em seguida, realizamos o processo de adaptação, que consiste em unir os dois anéis r e t, compondo um único anel r. Caso a união dos anéis r e t não seja viável, apenas marcamos a aresta (u, v) como selecionada.

No Algoritmo 2, apresentamos o algoritmo informal para a fase de construção descrita anteriormente. No passo 1, inicializamos a solução (inicial) S. No passo 2, criamos o conjunto W de arestas não selecionadas. Nos passos 3-17, tentamos diminuir o número de anéis da solução S através da união de dois anéis distintos. No passo 5, obtemos a lista LRC a partir do conjunto W. No passo 6, seleciona-se as β melhores arestas candidatas, a partir da lista LRC. No passo 7, seleciona-se, de forma aleatória, uma aresta $(u, v) \in LRC$, tal que u pertence ao anel r e v pertence ao anel t. Nos passos

Algoritmo 2 Algoritmo informal para a fase de construção do *GRASP*

1: Crie uma solução S onde cada localidade $u \in V$ formará um anel em S
2: Crie o conjunto W de arestas não selecionadas, ou seja, $W := E$
3: **Enquanto** $W \neq \emptyset$ **faça**
4: /* *construção da lista LRC* */
5: $LRC := W$
6: Aplique o algoritmo Estatística de Ordem na lista LRC
7: Escolha uma aresta $(u, v) \in LRC$ de forma aleatória com $u \in r$ e $v \in t$
8: /* *adaptação* */
9: **Se** $r \cup t$ corresponde a um anel viável **Então**
10: /* *atualizando a solução* S */
11: **Para todo** $x \in t$ **faça**
12: $r := r \cup \{x\}$
13: $W := W \setminus \{x, y\}$ tal que $x \in r$ e $y \in t$
14: **Fim-Para**
15: **Senão**
16: $W := W \setminus \{(u, v)\}$
17: **Fim-Se**
18: **Fim-Enquanto**

9-16, é realizado o processo de adaptação. No passo 9, verificamos se a união dos anéis r e t é viável, ou seja, se a capacidade do novo anel é menor ou igual a B. Se isto ocorrer, unimos os dois anéis e marcamos as arestas entre esses dois anéis como arestas já selecionadas. Caso contrário, a união não será feita, e marcamos apenas a aresta (u, v) como selecionada.

4.2.2 Fase de busca local

Na fase de busca local, as soluções vizinhas à solução obtida na fase de construção serão geradas, utilizando-se os movimentos de melhorias empregados por Lin e Kernighan para problemas de particionamento de grafos (vide [34]).

Dada uma solução viável S, tentamos iterativamente melhorar o valor da função objetivo (4.7). Esses melhoramentos podem ser obtidos através de dois procedimentos de perturbação. São eles:

Pr1: considere dois anéis r e t da solução S. Uma localidade u pertencente ao anel r é movida para o anel t;

Pr2: nesse segundo movimento, realizamos uma troca de localidades entre

dois anéis, ou seja, uma localidade u pertencente ao anel r é movida para o anel t e uma localidade v pertencentes ao anel t é movida para o anel r.

Estas trocas possibilitam a formação de soluções vizinhas à solução corrente S. Em ambos os movimentos Pr1 e Pr2, a solução corrente é atualizada, à medida que se encontra uma solução melhor na sua vizinhança. Os passos envolvidos nesta fase estão apresentados no algoritmo informal abaixo. Neste algoritmo, assuma que Ind_i e Ind_j representam os anéis, e que Ind_u e Ind_v representam as localidades onde foi possível realizar uma troca com alguma melhoria na solução.

Algoritmo 3 Algoritmo informal para a fase de busca local do *GRASP*

```
1: Seja S a solução (inicial) obtida na fase de contrução
2: Seja z o valor da função objetivo associado à solução S
3:
4: /* procedimento de perturbação Pr1 */
5: Para cada anel i de S faça
6:    Para cada anel j com i ≠ j faça
7:       Escolha uma localidade u ∈ i
8:       i := i \ {u}; j := j ∪ {u}
9:       Se foi possível melhorar o valor de z Então
10:         Atualize o valor de z
11:         Ind_i := i; Ind_j := j; Ind_u := u
12:      Fim-Se
13:   Fim-Para
14:   Atualize S, observando os índices Ind_i, Ind_j e Ind_u
15: Fim-Para
16:
17: /* procedimento de perturbação Pr2 */
18: Para cada anel i de S faça
19:    Para cada anel j com i ≠ j faça
20:       Escolha um par de localidades u ∈ i e v ∈ j, tal que, v ≠ u
21:       i := i \ {u}; j := j ∪ {u}
22:       i := i ∪ {v}; j := j \ {v}
23:       Se foi possível melhorar o valor de z Então
24:          Atualize o valor de z
25:          Ind_i := i; Ind_j := j; Ind_u := u; Ind_v := v
26:       Fim-Se
27:    Fim-Para
28:    Atualize S, observando os índices Ind_i, Ind_j, Ind_u e Ind_v
29: Fim-Para
```

Um algoritmo informal para a metaheurística *GRASP* que propusemos para o problema de atribuição de localidades a anéis é apresentado em seguida.

Na próxima seção, apresentaremos os resultados computacionais obtidos com a execução do *GRASP* para algumas instâncias do SRAP testadas.

Algoritmo 4 Algoritmo informal para a metaheurística *GRASP*

1: Seja z o valor do limitante superior obtido em cada iteração
2: Seja z_{ub} o valor do melhor limitante superior
3: **Enquanto** o número de iterações não for atingido **faça**
4: *GRASP*-SRAP-construção(S,z)
5: *GRASP*-SRAP-busca-local(S,z)
6: **Se** S apresenta alguma melhoria **Então**
7: Atualize o valor de z_{ub}, ou seja, $z_{ub} := z$
8: **Fim-Se**
9: **Fim-Enquanto**

4.3 Resultados computacionais

Nesta seção, apresentamos os resultados computacionais obtidos com a metaheurística *GRASP* descrita anteriormente. Esta metaheurística foi implementada utilizando a linguagem `C`. Todos os experimentos computacionais foram realizados em um computador com processador `AMD K6 450MHz` e com `256 MBytes` de memória `RAM`.

Para investigarmos o desempenho da metaheurística *GRASP*, utilizamos as instâncias pertencentes às classes C1 e C2, descritas na seção 2.3. Estas classes de instâncias foram divididas de acordo com a forma como foram geradas - geométricas ou aleatórias - e segundo a capacidade de um anel - $B = 155Mbs$ ou $B = 622Mbs$.

Antes de apresentarmos os resultados obtidos, mencionaremos os valores adotados para os parâmetros da metaheurística *GRASP*. Estabeleceu-se que o número máximo de iterações, denotado por NUM_MAX_ITER, fosse igual a duas vezes o número de vértices. Os valores escolhidos para β foram $\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$.

Vale ressaltar que o objetivo de termos implementado uma metaheurística *GRASP* é apenas o de fornecer um bom limitante superior inicial para um

algoritmo *branch-and-bound*. Assim, nesta seção, não nos preocupamos em mencionar o tempo computacional utilizado na obtenção destes limitantes. Entretanto, a título de curiosidade, informamos ao leitor que esse tempo computacional, em média, não ultrapassa 10 segundos para as instâncias testadas.

4.3.1 Classe C1

Essa primeira classe de instâncias, como dito anteriormente, foi gerada por Goldschmidt *et al.* [26] e empregada por esses autores para testar as suas heurísticas gulosas. Informamos nossos resultados apenas para 111 das 160 instâncias pertencentes a esta classe, visto que aqueles autores provaram que as demais 49 instâncias eram inviáveis.

Através das heurísticas gulosas, Goldschmidt *et al.* conseguiram obter o valor ótimo nas 111 instâncias viáveis. Um resumo desses resultados é apresentado na Tabela 4.1. Na primeira coluna, indicamos o grupo das instâncias; na segunda coluna, indicamos a capacidade de um anel e, por último, as colunas com as cardinalidades. Para cada cardinalidade, indicamos o número de instâncias onde foi possível encontrar a solução ótima.

Grupo	Capacidade (B)	Cardinalidade ($\|V\|$)			
		15	25	30	50
geométrica	155Mbs	4	5	9	5
geométrica	622Mbs	8	7	7	5
aleatória	155Mbs	7	8	6	7
aleatória	622Mbs	7	10	9	7

Tabela 4.1: Resultados obtidos com as heurísticas gulosas para as instâncias da classe C1.

Na Tabela 4.2, apresentamos os resultados computacionais obtidos com a metaheurística *GRASP* para 111 instâncias viáveis da classe C1. Para cada cardinalidade, temos o número de instâncias com aquela cardinalidade onde foi possível encontrar o valor ótimo pela metaheurística *GRASP*.

Foi possível encontrar o ótimo inteiro em 101 das 111 instâncias testadas. Nestas instâncias, os melhores valores de β foram iguais a 2, 3 e 5. Nas 10 instâncias restantes, encontramos soluções viáveis que distam no máximo

Grupo	Capacidade (B)	Total	*GRASP*				
			Cardinalidade ($\|V\|$)				Total
			15	25	30	50	
geométrica	155Mbs	23	4	5	9	3	21
geométrica	622Mbs	27	8	6	7	4	25
aleatória	155Mbs	28	7	8	6	3	24
aleatória	622Mbs	33	7	10	9	5	31

Tabela 4.2: Resultados obtidos com a metaheurística *GRASP* para as instâncias da classe C1.

33.33% do ótimo inteiro. Na Tabela 4.3, apresentamos os valores obtidos para estas 10 instâncias. A primeira coluna indica o nome da instância testada; a segunda coluna indica o limite inferior z_{lb}. Esse valor pode ser calculado por

$$z_{lb} = \lceil D/B \rceil, \tag{4.8}$$

onde $D = \sum_{u,v \in E} d_{uv}$ corresponde à demanda total. A terceira coluna indica o valor ótimo z^*; a quarta coluna, o valor z obtido pela metaheurística *GRASP*, e a quinta coluna, a diferença percentual entre o valor ótimo e o valor obtido pelo *GRASP*:

$$\Delta(z_{ub}, z^*) = \frac{z_{ub} - z^*}{z_{ub}} \times 100. \tag{4.9}$$

Por último, a sexta e sétima colunas indicam, respectivamente, o número da iteração e o valor de β, onde encontramos o valor z_{ub} para estas soluções.

4.3.2 Classe C2

Nessa segunda classe, foram utilizadas as instâncias geradas por Aringhieri e Dell'Amico [3]. Esses autores geraram 230 instâncias viáveis. Neste segundo experimento, informamos nossos resultados para todas essas instâncias. Um resumo, com uma pequena descrição sobre elas, é apresentado na Tabela 4.4.

Aringhieri e Dell'Amico conseguiram obter o valor ótimo de todas as instâncias apresentadas na Tabela 4.4, com o emprego da metaheurística busca tabu descrita na seção 4.1.2.

Instância			*GRASP*			
Nome	z_{lb}	z^*	z_{ub}	$\Delta(z_{ub}, z^*)$	# Iter	β
gs_50_1	4	5	6	16.67	2	2
gs_50_8	5	5	6	16.67	2	2
gl_25_2	2	2	3	33.33	1	2
gl_50_1	4	5	6	16.67	10	2
rs_50_3	4	4	5	20	2	2
rs_50_4	4	4	5	20	17	2
rs_50_5	4	4	5	20	4	2
rs_50_10	3	4	5	20	1	2
rl_50_2	3	3	4	25	1	2
rl_50_10	4	4	5	20	40	2

Tabela 4.3: Soluções viáveis obtidas com a metaheurística *GRASP* para algumas instâncias da classe C1.

Grupo	Capacidade (B)	Cardinalidade ($\|V\|$)			
		15	25	30	50
geométrica	155Mbs	20	20	10	20
geométrica	622Mbs	0	10	10	50
aleatória	155Mbs	20	10	20	20
aleatória	622Mbs	10	0	10	0

Tabela 4.4: Distribuição das instâncias da classe C2 testadas com a metaheurística *GRASP*.

A Tabela 4.5 apresenta os resultados computacionais obtidos com o emprego da metaheurística *GRASP*. Observa-se que foi possível encontrar o valor ótimo em 136 das 230 instâncias.

Grupo	Capacidade (B)	Total	*GRASP*				
			Cardinalidade ($\|V\|$)				Total
			15	25	30	50	
geométrica	155Mbs	70	20	20	8	0	48
geométrica	622Mbs	70	0	10	9	6	25
aleatória	155Mbs	70	19	10	14	0	43
aleatória	622Mbs	20	10	0	10	0	20

Tabela 4.5: Resultados obtidos com a metaheurística *GRASP* para as instâncias da classe C2.

Relatamos, em seguida, o que ocorreu nos experimentos computacionais para as 94 instâncias onde não foi possível encontrar o valor ótimo. Em 31 daquelas instâncias, encontramos soluções viáveis, que distam no máximo 28.57% do valor ótimo. Essas soluções viáveis foram encontradas para valores de β iguais a 2 e 5. Nas demais 63 instâncias não foi possível encontrar uma solução viável. As duas colocações anteriores decorrem da análise dos resultados contidos nas Tabelas 4.6 e 4.7, respectivamente.

4.4 Considerações

Através dos resultados apresentados, para ambas as classes de instâncias C1 e C2, nota-se que na maioria das instâncias testadas foi possível obter limitantes superiores para um algoritmo *branch-and-bound*.

No próximo capítulo, veremos como usar estes limitantes superiores, na tentativa de melhorar o desempenho computacional de um algoritmo *branch-and-price* empregado na resolução do problema de atribuição de localidades a anéis.

Instância			*GRASP*			
Nome	z_{lb}	z^*	z_{ub}	$\Delta(z_{ub}, z^*)$	# Iter	β
gl_30_10.2	4	5	6	16.67	5	3
gl_30_10.5	4	5	6	16.67	14	6
gl_50_6.1	5	6	8	25	81	2
gl_50_6.3	5	6	7	14.29	56	3
gl_50_6.6	5	6	7	14.29	15	5
gl_50_10.1	4	5	6	16.67	8	5
gl_50_10.2	4	5	6	16.67	17	2
gl_50_10.3	4	6	7	14.29	105	2
gl_50_10.4	4	5	6	16.67	2	4
gl_50_10.5	4	5	7	28.57	73	2
gl_50_10.6	4	5	7	28.57	80	2
gl_50_10.8	4	5	6	16.67	6	3
gl_50_10.9	4	6	7	14.29	93	3
gh_30_8.2	3	4	5	20	1	4
gh_50_2.6	5	6	8	25	25	3
gh_50_2.8	5	6	7	14.29	88	5
gh_50_3.1	4	5	6	16.67	10	3
gh_50_3.2	4	5	8	37.50	16	2
gh_50_3.4	4	5	6	16.67	53	4
gh_50_3.5	4	5	6	16.67	52	3
gh_50_3.8	4	5	6	16.67	37	5
gh_50_3.9	4	5	6	16.67	22	4
gh_50_9.1	4	5	7	28.57	103	5
gh_50_9.2	4	5	6	16.67	5	5
gh_50_9.3	4	5	6	16.67	79	5
gh_50_9.8	4	5	6	16.67	26	3
gh_50_9.10	4	5	6	16.67	20	3
rl_15_2.7	2	3	4	25	1	5
rl_30_2.1	3	4	5	20	73	5
rl_30_2.6	3	4	5	20	1	4
rl_30_2.10	3	4	5	20	116	2

Tabela 4.6: Soluções viáveis obtidas com a metaheurística *GRASP* para algumas instâncias da classe C2.

Grupo	Capacidade (B)	Cardinalidade ($\|V\|$)			
		15	25	30	50
geométrica	155Mbs	0	0	0	9
geométrica	622Mbs	0	0	0	31
aleatória	155Mbs	0	0	3	20
aleatória	622Mbs	0	0	0	0

Tabela 4.7: Distribuição das instâncias da classe C2, onde a metaheurística *GRASP* não obteve soluções viáveis.

Capítulo 5

Algoritmo *branch-and-price* para o problema de atribuição de localidades a anéis

Nesse capítulo, descrevemos o algoritmo *branch-and-price* que implementamos para a resolução do problema de atribuição de localidades a anéis. Esta técnica algorítmica vem sendo aplicada nos últimos anos na resolução de alguns problemas de otimização combinatória, como por exemplo, em problemas de roteamento e escalonamento (vide Sol e Savelsbergh [67]), em problemas de *bin packing* e *cutting stock* (vide Vanderbeck [72], em problemas de telecomunicações (vide Maculan *et al.* [48]), e em outros problemas (vide Hansen *et al.* [28], Jeong *et al.* [30] e Savelsbergh [64]).

Iniciamos este capítulo, com uma breve introdução sobre alguns aspectos em que se baseia o nosso algoritmo, levantando alguns pontos relevantes para o desempenho de qualquer algoritmo *branch-and-price*.

Apresentamos, ainda, uma reformulação do problema de atribuição de localidades a anéis como sendo um problema de *set partitioning*. Salientamos, também, a estrutura do problema mestre e do problema auxiliar e descrevemos um estudo inicial sobre a estrutura facial do politopo associado ao problema auxiliar.

Por último, descrevemos os detalhes de implementação do algoritmo *branch-and-price*: estratégias de ramificação, procedimentos de *bounding*, seleção de nós, processo de geração de colunas, etc. Em seguida, apresentamos os resultados computacionais obtidos com o emprego deste algoritmo na

resolução do problema de atribuição de localidades a anéis.

5.1 Introdução

A exemplo dos algoritmos planos-de-corte, que podem ser projetados conjuntamente com procedimentos *branch-and-bound*, os algoritmos de geração de colunas podem ser combinados com um método de enumeração implícita.

Esta combinação de procedimentos *branch-and-bound* com geração de colunas foi implementada pela primeira vez por Desrosiers *et al.* [17] para um problema de roteamento de veículos com janelas de tempo. A denominação algoritmo *branch-and-price* foi dada por Martin Savelsbergh, em analogia à nomenclatura *branch-and-cut* (veja Thienel [71], página 37).

O "coração" de um algoritmo *branch-and-price* está na geração de colunas para o problema mestre. Esse processo de geração de colunas é feito por um problema, denominado problema auxiliar, que tenta verificar a otimalidade de um problema linear (PL), associado ao problema mestre, em cada nó da árvore *branch-and-bound.*

À primeira vista, pode-se pensar que um algoritmo *branch-and-price* envolva apenas a combinação de duas técnicas bastante conhecidas: geração de colunas e procedimentos *branch-and-bound.* Porém, o emprego de uma técnica desenvolvida para programação linear com um método desenvolvido para programação linear inteira pode acarretar algumas dificuldades (vide Appelgren [2] e Johnson [31]):

- o emprego de estratégias de ramificação para programação linear inteira pode destruir a estrutura do problema auxiliar,
- e com isso, a prova da otimalidade dos programas lineares fica prejudicada, devido à resolução ineficiente dos problemas auxiliares.

Essas duas dificuldades podem ser amenizadas com o emprego adequado de algumas regras de ramificação propostas por Ryan e Foster [63] e, mais recentemente, por Vanderbeck [73].

Podemos, ainda, mencionar três fatores, enumerados por Barnhart *et al.* [8], que sugerem diretamente do emprego de algoritmos *branch-and-price* para a resolução de problemas de otimização combinatória, caso a formulação natural definida para este problema apresente um dos fatores abaixo:

1. os limitantes duais obtidos através da formulação natural do problema como um problema de programação inteira (mista) são ruins;

2. a formulação natural do problema como um problema de programação inteira (mista) possui uma estrutura com simetrias que provoca um desempenho pobre para o procedimento *branch-and-bound*;

3. a técnica de geração de colunas possibilita a decomposição do problema de programação linear em dois problemas: o problema mestre e o problema auxiliar.

Conforme discutido no Capítulo 3, os fatores 1 e 2 enumerados acima estão presentes nas formulações apresentadas até aqui. Portanto, justifica-se a aplicação da técnica de geração de colunas e de um algoritmo *branch-and-price* para tratar o SRAP.

Antes de passarmos à formulação do SRAP como um problema de *set partitioning*, apresentamos na próxima seção uma reformulação do problema de atribuição de localidades a anéis, a qual será usada como base para a nova modelagem. Ainda nesta seção, fazemos uma comparação entre os limitantes obtidos por essa nova formulação e pela formulação (P2).

Na seção 5.3, descrevemos o modelo de particionamento para o SRAP, de acordo com a técnica básica de geração de colunas, ou seja, definimos os problemas mestre e auxiliar. Na seção 5.4, apresentamos um estudo poliédrico preliminar do problema auxiliar.

Na seção 5.5, descrevemos as componentes básicas do algoritmo *branch-and-price* que foi implementado e alguns resultados computacionais usados para ajustá-lo. Na seção 5.6 são reportados os experimentos computacionais que realizamos com o algoritmo *branch-and-price* e, comparamos estes resultados com aqueles obtidos pelos modelos dos capítulos anteriores.

5.2 Reformulando o SRAP

Nesta seção, apresentamos uma reformulação do problema de atribuição de localidades a anéis. Seja A uma solução para o SRAP, ou seja, uma partição viável contendo os agrupamentos $V_1, \ldots, V_l$. Considere, ainda, os seguintes valores obtidos a partir desta solução:

$$D = \sum_{u} \sum_{u<v} d_{uv}, \tag{5.1}$$

$$D_j = \sum_{u \in V_j} \sum_{\substack{v \in V_j \\ u<v}} d_{uv}, \text{ para todo } j = 1, \ldots, l, \tag{5.2}$$

e

$$\overline{D} = D - \sum_{j=1}^{l} D_j, \tag{5.3}$$

ou seja, D é a demanda total e D_j representa a soma das demandas dentro do agrupamento j.

Agora, a soma das demandas nas arestas do corte V_j é dada por

$$W_j = \sum_{u \in V_j} \sum_{\substack{v \notin V_j \\ u<v}} d_{uv}. \tag{5.4}$$

Assim, tem-se que

$$2 \times \overline{D} = \sum_{j=1}^{l} W_j. \tag{5.5}$$

Logo, a restrição de capacidade do agrupamento federal obriga que

$$\begin{aligned} &\overline{D} \leq B \Longrightarrow D + \overline{D} \leq D + B \Longrightarrow \\ &\overset{(5.3)}{\Longrightarrow} \overline{D} + \overline{D} + \sum_{j=1}^{\ell} D_j \leq D + B \Longrightarrow \\ &\overset{(5.5)}{\Longrightarrow} \sum_{j=1}^{\ell} W_j + \sum_{j=1}^{\ell} D_j \leq D + B. \end{aligned} \tag{5.6}$$

Através dos valores definidos acima, podemos agora apresentar uma reformulação do SRAP como sendo um problema de programação linear inteira 0-1. Para tanto, considere a definição das seguintes variáveis de decisão:

$$x_{ui} = \begin{cases} 1, & \text{se a localidade } u \text{ é atribuída ao agrupamento } i. \\ 0, & \text{caso contrário.} \end{cases}$$

$$y_i = \begin{cases} 1, & \text{se o agrupamento } i \text{ está ativo.} \\ 0, & \text{caso contrário.} \end{cases}$$

$$f_{uvi} = \begin{cases} 0, & \text{se ambas as localidades } u \text{ e } v \text{ não pertencem ao agrupamento } i. \\ 1, & \text{caso contrário.} \end{cases}$$

onde f_{uvi} representa o vetor de incidência das arestas induzidas por V_i ou que estejam no corte $\delta(V_i)$.

Usando as variáveis de decisão definidas acima, o problema de atribuição de localidades a anéis pode ser expresso através da seguinte formulação abaixo:

(P3)

$$\text{maximize } \sum_{u=1}^{n}\sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i,$$

sujeito a:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv} f_{uvi} \leq B, \qquad \forall i \in V, \tag{5.7}$$

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n}\sum_{i=1}^{n} d_{uv} f_{uvi} \leq D + B, \tag{5.8}$$

$$\sum_{i=1}^{n} x_{ui} \leq 1, \qquad \forall u \in V,$$

$$x_{ui} - y_i \leq 0, \qquad \forall u \in V, i \in V,$$

$$f_{uvi} - x_{ui} \geq 0, \qquad \forall u, v \in V, u < v, \ \forall i \in V, \tag{5.9}$$

$$f_{uvi} - x_{vi} \geq 0, \qquad \forall u, v \in V, u < v, \ \forall i \in V, \tag{5.10}$$

$$f_{uvi} - x_{ui} - x_{vi} \leq 0, \qquad \forall u, v \in V, u < v, \ \forall i \in V, \tag{5.11}$$

$$x_{ui} \in \{0, 1\}, \qquad \forall u \in V, \ \forall i \in V,$$

$$y_i \in \{0, 1\}, \qquad \forall i \in V,$$

$$f_{uvi} \in \{0, 1\}, \qquad \forall u, v \in V, u < v, \ \forall i \in V. \tag{5.12}$$

Em seguida, fazemos uma descrição de algumas das restrições presentes nesta formulação, algumas já explicadas no Capítulo 2. As restrições (5.7) estabelecem que o tráfego total de cada anel deve ser no máximo igual a B. A restrição (5.8) estabelece que a capacidade do anel federal será no máximo igual a $D + B$. As restrições (5.9) estabelecem que se a localidade u pertence ao agrupamento i, então f_{uvi} será a igual 1. O mesmo sentido pode ser estabelecido com relação às restrições (5.10), porém observando a localidade v. As restrições (5.11) estabelecem que se ambas as localidades u e v não pertencem ao agrupamento i, então f_{uvi} será a igual 0. As restrições (5.12) correspondem às restrições de integralidade.

Pode-se notar que esta formulação faz uso do princípio de empacotamento definido no Capítulo 2. O número de variáveis na formulação (P3) é igual a $n^2 + mn + n$ variáveis de decisão, enquanto que o número de restrições é $n^2 + 3mn + 2n + 1$. Comparando-se as formulações (P2) e (P3), nota-se que tivemos uma diminuição de $2mn$ variáveis de decisão.

A seguir realizamos uma comparação entre os limitantes inferiores obtidos pelas formulações (P2) e (P3) que propusemos neste trabalho. Até este ponto do trabalho, a formulação mais adequada à resolução do SRAP é a formulação (P2).

Procuramos estabelecer essas comparações, tomando por base os dois critérios empregados anteriormente no Capítulo 2: o número de nós abertos na árvore *branch-and-bound* e o tempo computacional. Ressaltamos que a formulação (P3) não conseguiu melhorar o valor da relaxação linear, que continuou sendo igual a 1 para todas as instâncias testadas.

Nas Tabelas 5.1 e 5.2, relatamos os resultados obtidos com algumas instâncias descritas no Capítulo 2. As tabelas apresentadas nesta seção estão

divididas em duas partes. Na primeira parte, temos as colunas `Nome` e z^*, que indicam, respectivamente, a instância testada e o valor do ótimo inteiro. A segunda parte indica os resultados obtidos com as formulações (P2) e (P3). A coluna `# Nds` indica o número de nós abertos na árvore *branch-and-bound*, a coluna z indica o limite inferior obtido por uma das formulações, a coluna Δ indica a diferença percentual entre o ótimo inteiro e o limitante obtido e, por último, a coluna `Tmp` indica o tempo computacional obtido por uma das formulações em segundos.

A Tabela 5.1 apresenta os resultados computacionais para as instâncias geométricas da classe C1. Das 14 instâncias testadas, podemos observar que a formulação (P3) provou a otimalidade em todas elas, antes de atingir o tempo computacional máximo, ou seja, antes de 18,000 segundos.

Pode-se verificar, ainda, que em apenas uma instância, gh_25_8, a formulação (P3) teve que abrir mais nós na árvore *branch-and-bound* do que a formulação (P2). Isso, entretanto, não ocasionou um tempo computacional maior para a resolução dessa instância.

Um outro aspecto positivo que podemos ressaltar com relação à formulação (P3) é que para todas as instâncias testadas na Tabela 5.1, o tempo computacional ficou abaixo de 25 minutos e, conseqüentemente, inferior ao obtido com o emprego da formulação (P2). Em algumas instâncias este tempo chegou a ser, no mínimo, 9 vezes (gl_15_1) e, no máximo, 40 vezes (gl_15_7) melhor do que o tempo computacional obtido pela formulação (P2), para instâncias de pequeno porte ($n = 15$). Já para as instâncias de médio porte ($n = 25$), temos que a melhoria de tempo chegou a ser de, no mínimo, 2 vezes (gh_25_4) e, no máximo, 17 vezes (gh_25_3). Note, ainda, que em 28.57% das instâncias testadas o tempo computacional gasto pela formulação (P3) foi inferior a 1 minuto.

A Tabela 5.2 apresenta os resultados obtidos para as instâncias aleatórias da classe C1. Foram testadas 19 instâncias. A formulação (P3) conseguiu provar a otimalidade em todas elas, antes de 18,000 segundos.

Note que em apenas 6 instâncias, a formulação (P3) teve que abrir mais nós na árvore *branch-and-bound* do que a formulação (P2). A única instância

onde o tempo computacional foi um pouco maior do que o obtido pela formulação (P2) deu-se com a resolução da instância rh_25_10. Acreditamos que isto se deva à quantidade de nós abertos na resolução desta instância.

Por último, vale ressaltar que em 89.47% das instâncias, a formulação (P3) apresentou um tempo computacional inferior a 25 minutos. Note que, para algumas instâncias, o fator de melhoria no tempo computacional chegou a ser 46 vezes melhor (rl_15_9), no caso de pequenas instâncias ($n = 15$), e de 8 vezes melhor (rh_25_9), no caso das instâncias de médio porte ($n = 25$). Além disso, podemos mencionar que os tempos computacionais obtidos pela formulação (P3) ficaram abaixo de 1 minuto em 47.37% das instâncias.

Na Tabela 5.3, apresentamos os resultados obtidos para algumas instâncias aleatórias da classe C2. Foram testadas 7 instâncias. Pode-se observar que a formulação (P3) também provou a otimalidade de todas elas, antes de atingir o tempo computacional máximo.

Em apenas 3 instâncias pertencentes a esta classe, new.rl_15_2.1, new.rh_15_10.2 e new.rh_15_10.6, a formulação (P3) teve que abrir mais nós do que a formulação (P2). Entretanto, a exemplo das instâncias anteriores, isso não ocasionou, diretamente, um aumento no tempo computacional.

Mais uma vez, o tempo computacional obtido pela formulação (P3), para as 7 instâncias testadas desta classe, foi menor do que o obtido pela formulação (P2). Em particular, esse tempo computacional ficou abaixo de 25 minutos para todas elas. Nota-se que esta melhoria no tempo computacional chegou a ser, no mínimo, de 4 vezes (vide rh_15_2.1) e, no máximo, 238 vezes (vide rl_15_5.1).

Assim, o desempenho computacional da formulação (P3) superou o desempenho da formulação (P2) nos experimentos computacionais realizados nesta seção. Devido a este sucesso, utilizamos a formulação (P3) como base para a modelagem que empregaremos na técnica de geração de colunas. Esta modelagem é apresentada na próxima seção.

		Formulação (P2)				Formulação (P3)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
gl_15_1	3	4873	3	0%	654	1870	3	0%	72
gl_15_4	3	3994	3	0%	665	1544	3	0%	32
gl_15_7	3	9838	3	0%	1880	1405	3	0%	46
gl_15_9	3	16030	3	0%	1875	5499	3	0%	69
gh_15_1	3	11954	3	0%	2222	8476	3	0%	156
gh_15_2	3	7424	3	0%	2296	2259	3	0%	106
gh_15_8	3	3645	3	0%	1731	2004	3	0%	171
gh_15_9	3	12357	3	0%	2599	1758	3	0%	68
gh_25_1	3	3202	3	0%	2481	2283	3	0%	723
gh_25_2	2	923	2	0%	451	376	2	0%	52
gh_25_3	2	1233	2	0%	689	288	2	0%	39
gh_25_4	3	8466	3	0%	2942	2583	3	0%	989
gh_25_8	3	2781	3	0%	4325	3639	3	0%	1327
gh_25_9	3	7420	3	0%	10261	1836	3	0%	780

Tabela 5.1: Comparação entre as formulações (P2) e (P3) para as instâncias geométricas da classe C1.

		Formulação (P2)				Formulação (P3)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
rl_15_1	3	9118	3	0%	1066	1917	3	0%	24
rl_15_3	2	168	2	0%	3	53	2	0%	1
rl_15_4	3	5752	3	0%	1201	1990	3	0%	53
rl_15_6	3	12548	3	0%	1816	4503	3	0%	59
rl_15_8	3	9591	3	0%	1775	4246	3	0%	63
rl_15_9	3	15270	3	0%	1730	3084	3	0%	37
rl_15_10	3	8853	3	0%	1825	1965	3	0%	64
rl_25_2	3	4869	3	0%	4037	4776	3	0%	848
rl_25_8	3	4044	3	0%	2136	14363	3	0%	1450
rh_15_1	3	4361	3	0%	1450	4417	3	0%	114
rh_15_4	2	85	2	0%	6	52	2	0%	1
rh_15_5	3	9039	3	0%	2378	9093	3	0%	203
rh_15_6	3	16800	3	0%	3827	6648	3	0%	119
rh_15_7	2	187	2	0%	10	143	2	0%	1
rh_15_8	2	546	2	0%	34	660	2	0%	8
rh_15_9	3	18319	3	0%	3629	7724	3	0%	140
rh_25_7	3	3103	3	0%	4588	33740	3	0%	3867
rh_25_9	2	353	2	0%	77	93	2	0%	9
rh_25_10	3	14037	3	0%	13870	160709	3	0%	15650

Tabela 5.2: Comparação entre as formulações (P2) e (P3) para as instâncias aleatórias da classe C1.

		Formulação (P2)				Formulação (P3)			
Nome	z^*	#Nds	z	Δ	Tmp	#Nds	z	Δ	Tmp
new.rl_15_2.1	3	5783	3	0%	858	10744	3	0%	181
new.rl_15_2.3	3	9514	3	0%	1221	8281	3	0%	191
new.rl_15_5.1	3	17745	3	0%	6673	1113	3	0%	28
new.rl_15_5.2	3	15193	3	0%	4803	2552	3	0%	68
new.rl_15_5.3	3	3009	3	0%	981	2774	3	0%	62
new.rh_15_10.2	3	3421	3	0%	4113	7740	3	0%	716
new.rh_15_10.6	3	12533	3	0%	15542	13631	3	0%	1287

Tabela 5.3: Comparação entre as formulações (P2) e (P3) para as instâncias aleatórias da classe C2.

5.3 Um modelo de geração de colunas

Muitos problemas de otimização combinatória podem ser formulados como problemas de *set partitioning* (Balas e Padberg [6]). Como mostramos, a seguir, o problema mestre formulado para o problema de atribuição de localidades a anéis é um deles. Além disso, vários casos de sucesso na aplicação de algoritmos *branch-and-price* são reportados na literatura para este tipo de modelo. Para isto, contudo, é preciso definir o problema auxiliar, o que também será feito, a seguir, para o SRAP.

5.3.1 Definindo o problema mestre

Para formular o problema mestre como sendo um problema de *set partitioning*, definimos os seguintes valores:

- variável de decisão:

$$y_j = \begin{cases} 1, & \text{se } V_j \subseteq V \text{ forma um agrupamento viável.} \\ 0, & \text{caso contrário.} \end{cases}$$

- coluna gerada pelo problema auxiliar:

$$a_{uj} = \begin{cases} 1, & \text{se a localidade } u \in V_j. \\ 0, & \text{caso contrário.} \end{cases}$$

Assim, o problema mestre pode ser definido como

(P4)

$$\text{minimize} \sum_{j=1}^{K} y_j \tag{5.13}$$

sujeito a:

$$\sum_{j=1}^{K} a_{uj} y_j = 1, \qquad \forall u \in V, \tag{5.14}$$

$$\sum_{j=1}^{K} d_j y_j \leq D + B, \tag{5.15}$$

$$y_j \in \{0, 1\}, \qquad \forall j \in V,$$

onde $d_j = W_j + D_j$ para todo $j = 1, \ldots, n$.

As restrições (5.14) estabelecem que cada localidade u deve ser atribuída a um único agrupamento V_j. Já a restrição (5.15) estabelece que o tráfego total no anel federal é no máximo igual a $D + B$.

Como K pode apresentar um valor muito maior do que n, temos que a resolução da formulação (P4) pode ser computacionalmente custosa. Além disso, devido ao valor de K, a formulação (P4) apresenta mais do que n variáveis e $n + 1$ restrições.

Assim, uma maneira de resolvermos esta formulação de forma eficiente é gerando variáveis (ou colunas) y_j, com $1 \leq j \leq K$, através de um problema auxiliar. Na próxima seção, apresentamos o problema auxiliar encarregado de gerar agrupamentos viáveis V_j para o problema mestre definido acima.

5.3.2 Definindo o problema auxiliar

Considere um agrupamento viável V_j. Seja I_j o conjunto de vértices em V_j e, as variáveis duais μ_i e π associadas às restrições (5.14) e (5.15), respectivamente. Então, o custo reduzido do agrupamento V_j é dado por:

$$\bar{c}_j = 1 - \sum_{i \in I_j} \mu_i - \pi d_j. \tag{5.16}$$

Como o problema mestre é de minimização, o objetivo do problema auxiliar é encontrar um agrupamento V_j, tal que, $\bar{c}_j$ seja mínimo.

Considere, agora, as seguintes variáveis de decisão definidas para o agrupamento V_j:

$$x_i = \begin{cases} 1, & \text{se } i \in V_j. \\ 0, & \text{caso contrário.} \end{cases}$$

$$f_{ik} = \begin{cases} 0, & \text{se os vértices } i \text{ e } k \notin V_j. \\ 1, & \text{caso contrário.} \end{cases}$$

O problema auxiliar pode, então, ser formulado como o problema de programação linear inteira 0-1 abaixo. Note que as restrições do subproblema nada mais são do que as restrições da formulação (P4) para um agrupamento fixado, ou seja, para um agrupamento V_j.

Pelas restrições (5.18)-(5.20), temos que as localidades i e k podem ser atribuídas ao agrupamento V_j, se elas satisfazem a restrição de capacidade (5.21).

Observe que, se o valor da função (5.17) for maior do que 1, então podemos adicionar a coluna, correspondente ao agrupamento V_j, no problema mestre.

(P5)

$$\text{maximize } \sum_{i=1}^{n} \mu_i x_i + \pi \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} d_{ik} f_{ik}, \tag{5.17}$$

sujeito a:

$$f_{ik} - x_i \geq 0, \qquad \forall i, k \in V,\ i < k, \tag{5.18}$$

$$f_{ik} - x_k \geq 0, \qquad \forall i, k \in V,\ i < k, \tag{5.19}$$

$$f_{ik} - x_i - x_k \leq 0, \qquad \forall i, k \in V, i < k, \tag{5.20}$$

$$\sum_{i=1}^{n-1} \sum_{k=i+1}^{n} d_{ik} f_{ik} \leq B, \tag{5.21}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in V, \tag{5.22}$$

$$f_{ik} \in \{0, 1\}, \qquad \forall i, k \in V, i < k. \tag{5.23}$$

Esta formulação requer $n(n-1)/2+n$ variáveis de decisão e $3n(n-1)/2+1$ restrições. Observe que corresponde a fator da $O(n^2)$.

5.4 Estudo poliédrico do problema auxiliar

De acordo com a literatura, o Problema de Programação Quadrática Máxima é NP-difícil (veja Garey e Johnson [24]). Portanto, é possível demonstrar, através de uma redução polinomial, que o problema auxiliar também é NP-difícil.

Para a redução polinomial, podemos definir a função objetivo do problema auxiliar como sendo uma função quadrática, já que $f_{ik} = x_i + x_k - x_i * x_k$. Assim, podemos escrever o nosso problema como um problema quadrático 0-1 só nas variáveis x.

Após este argumento sobre a NP-completude do problema auxiliar, concluímos que seria interessante estudar a envoltória convexa das soluções inteiras do problema auxiliar, visto que a resolução do problema auxiliar pode ser computacionalmente "cara".

Os resultados apresentados nesta seção foram discutidos em de Souza [16]. Por este motivo, as provas dos teoremas estão omitidas.

Denotaremos por $\mathcal{P}'$ o politopo associado à formulação (P5). Assim, a definição de $\mathcal{P}'$corresponde a envoltória convexa de suas soluções inteiras, ou seja,

$$\mathcal{P}' = \; conv\,\{(x,f) \in \mathbb{B}^{n+n(n-1)/2} \mid (x,f) \text{ satisfaz } (5.18),(5.19),$$
$$(5.20),(5.21),(5.22) \text{ e } (5.23)\}.$$

do sistema (5.18)–(5.21), considerando que x_i e $f_{ik} \in \{0,1\}$, alguns resultados obtidos referentes à estrutura facial de $\mathcal{P}'$ são listados abaixo. Estes resultados assumem que qualquer par de vértices sempre forma um agrupamento viável, e que o grafo sobre o qual a instância do SRAP está definida é completo.

Teorema 5.1 *Para $n \geq 2$, a dimensão do politopo $\mathcal{P}'$ é cheia.*

Como o politopo $\mathcal{P}'$ possui dimensão cheia, então, cada faceta de $\mathcal{P}'$ é definida de forma única, a menos de uma multiplicação por um escalar positivo. Apresentaremos, em seguida, algumas desigualdades válidas e algumas

facetas para o politopo $\mathcal{P}'$. Inicialmente, procuraremos, sempre que possível, caracterizar quais dentre as desigualdades pertencentes à formulação (P4) são definidoras de facetas para o referido politopo.

Teorema 5.2 *Sejam i e k dois vértices distintos de V. Se $\{i,k,j\}$ é viável para todo $j \in V - \{i,k\}$, então a desigualdade (5.18) define uma faceta de $\mathcal{P}'$.*

Uma versão do resultado do teorema anterior pode ser dada para a desigualdade (5.19).

Teorema 5.3 *Sejam i e k dois vértices distintos de V. Se $\{i,k,j\}$ é viável para todo $j \in V - \{i,k\}$, então a desigualdade (5.19) define uma faceta de $\mathcal{P}'$.*

Uma versão mais geral do resultado dos Teoremas 5.2 e 5.3 é dada pelo seguinte resultado:

Teorema 5.4 *Seja k um vértice de V e $J = \{j_0, j_1, \ldots, j_p\}$ um subconjunto de $V - \{k\}$, tal que $\{k, j_q, j_r\}$ é um agrupamento inviável para todo $q \neq r$ com $(q,r) \in J \times J$. Então, a desigualdade (5.24) abaixo define uma faceta de $\mathcal{P}'$:*

$$x_{j_0} - f_{kj_0} + \sum_{j=1}^{p}(x_k + x_{j_q} - f_{kj_q}) \leq 0 \qquad (5.24)$$

O próximo resultado é com relação à desigualdade (5.20).

Teorema 5.5 *Para todo par de vértices distintos i e k de V, a desigualdade (5.20) define uma faceta de $\mathcal{P}'$.*

Com os resultados anteriores, pode-se mostrar que as desigualdades triviais, ou seja, aquelas que limitam os limites inferiores e superiores das variáveis em 0 e 1, respectivamente, não definem facetas de $\mathcal{P}'$. A única exceção é dada pelo resultado abaixo.

Teorema 5.6 *Sejam i e k dois vértices distintos de V tais que:*

(i) *para todo* j *em* $V - \{i,k\}$, *o agrupamento* $\{i,j,k\}$ *é viável,*

(ii) *para todo para* (j,ℓ) *em* $(V-\{i,k\}) \times (V-\{i,k\})$, *pelo menos um dos dois agrupamentos,* $\{i,j,\ell\}$ *ou* $\{k,j,\ell\}$, *é viável.*

Então, a desigualdade $f_{ik} \leq 1$ *define uma faceta de* $\mathcal{P}'$.

Quando a condição (i) do Teorema 5.6 não é satisfeita, pode-se mostrar que a desigualdade

$$f_{ij} + f_{ik} + f_{jk} - x_i - x_j - x_k \leq 1, \tag{5.25}$$

é satisfeita na igualdade. Tal desigualdade é denominada *desigualdade triangular.* Pelo resultado abaixo, esta desigualdade é definidora de faceta para $\mathcal{P}'$.

Teorema 5.7 *Sejam* i, j *e* k *três vértices distintos em* V. *Então, a desigualdade triangular define uma faceta de* $\mathcal{P}'$.

A desigualdade (5.25) é um caso particular de uma família de desigualdades, que chamamos de *desigualdades clique.* No teorema abaixo, apresentamos esta família de desigualdades e as condições segundo as quais definem facetas para o politopo $\mathcal{P}'$.

Teorema 5.8 *Seja* S *um subconjunto de* V *com* $|S| = 2c+1$ *para algum inteiro* $c \geq 1$. *Então, a desigualdade (5.26), dada por*

$$\sum_{e \in E(S)} f_e - c \sum_{i \in S} x_i \leq c(c+1)/2, \tag{5.26}$$

define uma faceta de $\mathcal{P}'$.

Observe que quando $c = 1$, obtemos a desigualdade triangular a partir da desigualdade clique.

Deve ser notado que, dentre todas as desigualdades apresentadas até aqui para o politopo $\mathcal{P}'$, a desigualdade (5.26) é a única que não aparece na formulação original do problema auxiliar.

Alguns testes computacionais preliminares realizado nos levaram a outra classe de desigualdades que podem melhorar os limitantes duais produzidos pelas relaxações lineares. Na verdade, estas desigualdades definem facetas do politopo $\mathcal{P}'$ e podem apresentar um melhor desempenho computacional, por estarem intimamente ligadas à restrição do tipo mochila, dada pela equação (5.21) da formulação do problema auxiliar. Estas desigualdades, denominadas *desigualdades árvore*, são introduzidas no teorema abaixo. Na oportunidade, mostramos condições necessárias para que estas desigualdades sejam definidoras de facetas para $\mathcal{P}'$.

Teorema 5.9 *Seja $(V(T), T)$ uma árvore em G, satisfazendo as seguintes propriedades:*

(i) *$V(T)$ não é um agrupamento viável e $V(T) - \{u\}$ é viável para qualquer vértice u em $V(T)$;*

(ii) *Para todo par de vértices não-adjacentes (u, v) em $V(T)$ e todo vértice z não pertencente a $V(T)$, sejam $V(T_u)$ e $V(T_v)$ os conjuntos de vértices das duas subárvores obtidas ao se remover o caminho que liga u a v em T e que contém os vértices u e v, respectivamente. O agrupamento formado por $V(T_u) \cup V(T_v) \cup \{z\}$ é viável;*

(iii) *Para todo u em V e todo z não pertencente a V, existe uma subárvore de G, cujo conjunto de vértices $V(T_u)$ contém u, tal que $V(T_u) \cup \{z\}$ é um agrupamento viável.*

Então, a desigualdade (5.27) dada abaixo

$$\sum_{u \in V(T)} x_u - \sum_{e \in T} f_e \leq 0, \tag{5.27}$$

define uma faceta de $\mathcal{P}'$.

A seguir, estabelecemos uma relação entre o politopo $\mathcal{P}'$ e alguns politopos associados a outros problemas de otimização combinatória bastante estudados na literatura. Para isso, mostraremos como derivar algumas desigualdades do politopo $\mathcal{P}'$, a partir de desigualdades daqueles politopos. A

iniciativa neste sentido foi provocada pelo fato de que a forma dos *grafos suporte* [1] das desigualdades definidoras de facetas dos teoremas anteriores assemelhava-se em muito àquele de desigualdades conhecidas para outros problemas combinatórios.

Estes grafos são análogos àqueles correspondentes a algumas das desigualdades definidoras de facetas dos politopos associados ao problema do corte ótimo de um grafo (CUT) e ao problema booleano quadrático (BQ). Um estudo sobre o politopo do problema CUT, denotado por $\mathcal{P}^{\mathcal{C}}$, pode ser encontrado em Barahona e Majhoub [7] e de Simone [15]. Já para o politopo do problema BQ, denotado por $\mathcal{P}^{\mathcal{B}}$ sugerimos Padberg [60] e Sherali *et al.* [65].

Assim, ao invés de continuarmos a busca por novas facetas de $\mathcal{P}'$, decidimos verificar se seria possível aproveitarmos os resultados já conhecidos para $\mathcal{P}^{\mathcal{C}}$ e $\mathcal{P}^{\mathcal{B}}$. Para entender o resultado alcançado, inicialmente formulamos os problemas $\mathcal{P}^{\mathcal{C}}$ e $\mathcal{P}^{\mathcal{B}}$ como programas lineares inteiros.

A formulação do problema CUT é dada pelo sistema (5.28) –(5.31), enquanto aquela do problema BQ é dada pelo sistema (5.32)–(5.34). Todas as variáveis das formulações abaixo são binárias.

$$x_i - x_j - z_{ij} \leq 0, \quad \forall (i,j) \in E, \tag{5.28}$$

$$x_j - x_i - z_{ij} \leq 0, \quad \forall (i,j) \in E, \tag{5.29}$$

$$z_{ij} + x_i + x_j \leq 2, \quad \forall (i,j) \in E, \tag{5.30}$$

$$z_{ij} - x_i - x_j \leq 0, \quad \forall (i,j) \in E, \tag{5.31}$$

$$y_{ij} - x_i \leq 0, \quad \forall (i,j) \in E, \tag{5.32}$$

$$y_{ij} - x_j \leq 0, \quad \forall (i,j) \in E, \tag{5.33}$$

$$x_i + x_j - y_{ij} \leq 1, \quad \forall (i,j) \in E. \tag{5.34}$$

No problema CUT, as variáveis z são associadas às arestas e fixadas em 1, se e somente se exatamente uma das suas extremidades pertence ao agrupamento[2]. Já no problema BQ, as variáveis y é que são associadas às arestas

[1] O grafo suporte de uma desigualdade é o subgrafo formado pelos vértices e arestas do grafo, cujas variáveis associadas têm coeficientes não-nulos naquela desigualdade.

[2] O termo agrupamento foi utilizado aqui para manter a compatibilidade com a definição do SRAP.

e tomam o valor 1, se e somente se ambas as suas extremidades estão no agrupamento.

Ao compararmos o sistema linear acima com a formulação do problema auxiliar dada por (5.18)–(5.20), verificamos que as seguintes relações lineares podem ser deduzidas para toda aresta (i,j) de E:

$$f_{ij} + y_{ij} = x_i + x_j, \tag{5.35}$$

$$f_{ij} - y_{ij} = z_{ij}, \tag{5.36}$$

$$z_{ij} = x_i + x_j - 2y_{ij}. \tag{5.37}$$

Como estas equações acima são lineares e estabelecem uma relação *um para um* entre as soluções dos três problemas considerados, toda desigualdade definidora de faceta em um dos três politopos correspondentes possui uma desigualdade definidora de faceta equivalente nos dois politopos restantes. Deve ser notado que a equação (5.37) já era conhecida na literatura e mostrava que vários resultados previamente publicados para $\mathcal{P}^{\mathcal{C}}$ e $\mathcal{P}^{\mathcal{B}}$ eram, na verdade, equivalentes.

Portanto, a descoberta das relações dadas pelas equações (5.35) e (5.36) foi importante, pois evitou que inadvertidamente pudessem ser considerados originais alguns dos resultados obtidos neste trabalho. De fato, após ter sido encontrado um outro artigo na literatura (veja Mehrotra [53]), no qual foi estudada a estrutura facial do politopo associado ao problema BQ, considerando uma restrição adicional do tipo *mochila*, concluímos que todas as desigualdades definidoras de facetas encontradas para o politopo $\mathcal{P}'$ associado ao problema auxiliar de geração de colunas do SRAP, já haviam sido propostas na literatura, em um outro contexto.

Com esta descoberta, tornou-se menos interessante, porém ainda atrativa, a continuidade do estudo poliedral do problema auxiliar. Neste caso, a direção que poderia ser tomada passaria pela implementação de um algoritmo do tipo *branch-and-cut*, que fizesse uso das desigualdades já conhecidas na literatura, para resolver o problema auxiliar de modo mais eficiente. Feito isto, só os resultados dos testes computacionais poderão indicar a necessi-

dade ou não de um estudo ainda mais aprofundado sobre a estrutura facial do politopo $\mathcal{P}'$. Contudo, veremos mais adiante, os experimentos realizados não apontaram para a necessidade de incluir planos-de-corte na resolução do problema auxiliar.

5.5 Visão geral do algoritmo *branch-and-price*

Nesta seção, apresentamos as decisões de implementação para o algoritmo *branch-and-price* que propusemos para a resolução do problema de atribuição de localidades a anéis.

O algoritmo *branch-and-price* que implementamos tem a sua estrutura baseada nos trabalhos de Barnhart *et al.* [8] e Vanderbeck e Wolsey [74], para problemas de programação inteira. Uma coletânea recente com aplicações de algoritmos *branch-and-price* na resolução de problemas de otimização combinatória pode ser vista em Maculan *et al.* [49].

A implementação de um algoritmo *branch-and-price* envolve muitas questões, por exemplo: como inicializar o algoritmo de geração de colunas (conjunto inicial de colunas), como resolver o problema auxiliar (algoritmo exato *versus* heurísticas), qual estratégia de seleção de colunas utilizar (múltiplas colunas ou uma única coluna a cada iteração do algoritmo de geração de colunas), qual estratégia empregar para a seleção dos nós, qual estratégia adotar para a fase de ramificação e quais procedimentos de *bounding* implementar?

A maioria dos algoritmos *branch-and-price* reportados na literatura faz uso de pacotes de otimização para resolução dos problemas lineares, como por exemplo, XPRESS 12.05 (vide [14]). Esse pacote oferece uma biblioteca de rotinas, denotada por XOSL, que facilita a implementação destes tipos de algoritmo.

Para que fosse possível o uso daqueles procedimentos, se fez necessário o desenvolvimento de uma estrutura de dados que fosse compatível com os procedimentos da biblioteca XOSL. Além disso, ainda desenvolvemos estruturas de dados que representassem convenientemente um grafo e uma árvore

de enumeração.

5.5.1 Conjunto inicial de colunas

O algoritmo *branch-and-price* precisa ser inicializado com uma solução viável para o problema mestre. Com o auxílio de uma heurística, é possível obtermos este conjunto inicial de colunas. Um algoritmo informal desta heurística é apresentado no Algoritmo 5. O valor CAP denota a capacidade máxima que um agrupamento poderá ter, ou seja, o valor B.

Além das colunas geradas pela heurística nos passos (2)-(39), incluímos também, no conjunto inicial de colunas, as colunas associadas aos agrupamentos canônicos, ou seja, os agrupamentos onde apenas x_{ii} é igual a 1. Estas colunas canônicas são geradas no passo 1. O custo associado a estas colunas é igual a $n+1$. Já a capacidade destes agrupamentos é igual a zero. Assim, temos garantias de que a heurística acima nos fornecerá uma base para o problema mestre.

Antes de continuarmos a descrição do algoritmo *branch-and-price*, faremos duas ressalvas sobre o conjunto inicial de colunas. Em primeiro lugar, há o risco de que estas colunas permaneçam na base do problema mestre por muito tempo; em segundo lugar, as colunas canônicas correm o risco de ficar no problema mestre ao final da resolução do primeiro nó da enumeração, significando que o problema é inviável. Entretanto, esta última não ocorre visto que as instâncias testadas correspondem a problemas viáveis.

5.5.2 Estrutura do procedimento *branch-and-bound*

Os principais aspectos de qualquer procedimento *branch-and-bound* para um problema de otimização combinatória são: estratégias de ramificação, seleção de nós e procedimentos de *bounding*. Em seguida, descreveremos cada um destes aspectos, e o que nos levou à escolha de um ou outro fator para a nossa implementação.

Algoritmo 5 Algoritmo informal para gerar um conjunto inicial de colunas

1: Gere as colunas canônicas
2: $NumTolCol := n$
3: $NumColIni := 5 \times n + 1$
4:
5: **Enquanto** $NumTotCol < NumColIni$ **faça**
6: /* *tamanho inicial do agrupamento que será gerado* */
7: Gere aleatoriamente um tamanho t, entre 2 e n, para o agrupamento
8:
9: /* *criando um agrupamento inicial com tamanho t* */
10: **Para** i = 1 até n **faça**
11: $permuta[i] := i$; /* *agrupamento inicial* */
12: **Fim-Para**
13:
14: /* *realizando algumas trocas no agrupamento inicial* */
15: /* *através destas trocas, obteremos o agrupamento final: agrup* */
16: $l := t$
17: **Para** i := 1 até t **faça**
18: Gere um índice j, entre 1 e l, de forma aleatória
19: $agrup[i] := permuta[j]$ /* *essa localidade foi escolhida* */
20: $permuta[j] := permuta[l - -]$
21: **Fim-Para**
22:
23: /* *calcule a capacidade do agrupamento gerado* */
24: $TI := \sum_{u,v \in agrup} d_{uv}$ /* *tráfego interno* */
25: $TE := \sum_{u \in agrup, v \notin agrup} d_{uv}$ /* *tráfego externo* */
26: $capac := TI + TE$
27:
28: /* *se o agrupamento gerado for inviável, tentamos torná-lo viável* */
29: **Enquanto** $capac > CAP$ **faça**
30: Escolha uma localidade $u \in agrup$ de forma aleatória
31: Marque a localidade u como não mais pertencente ao agrupamento *agrup*, ou seja, $agrup[i] := 0$
32: Calcule o valor de *capac* a partir do novo agrupamento *agrup*
33: **Fim-Enquanto**
34:
35: /* *verificando a existência de colunas* */
36: **Se** o agrupamento *agrup* ainda não foi gerado **Então**
37: $NumTotCol := NumTotCol + 1$
38: **Fim-Se**
39: **Fim-Enquanto**

Seleção de nós

Numa árvore *branch-and-bound* predominam dois tipos de nós: ativos e inativos. Um nó é considerado *ativo* se o seu processo de exploração ainda não foi finalizado. Por outro lado, diz-se que um nó é *inativo*, caso a sua exploração já tenha sido encerrada.

O processo de seleção é realizado sob os nós ativos. Caso não existam mais nós ativos, a melhor solução viável disponível no momento será o ótimo inteiro. Do contrário, devemos selecionar um nó ativo para prosseguir com a exploração. Três são as estratégias bem conhecidas na literatura: busca em profundidade (*depth-first search*), busca em largura (*breadth-first search*) e busca do melhor limitante (*best-first search*). Em seguida, mencionamos o motivo da escolha da estratégia do melhor limitante para o processo de seleção de nós.

Ambas as estratégias, busca em profundidade e em largura, possuem uma desvantagem crucial em relação à estratégia do melhor limitante. Durante o processo de seleção, podemos ter a busca direcionada a uma das partes da árvore *branch-and-bound* com limitantes inferiores locais ruins, o que diretamente se refletiria no tempo computacional. Tal desvantagem foi comprovada por Vanderbeck [75].

Devido a isto, a estratégia do melhor limitante foi a escolhida. Nesta estratégia, o nó a ser explorado será aquele considerado o mais promissor. Para o SRAP, isto corresponde ao nó cujo valor do limitante inferior local é o mínimo dentre todos os nós ativos. Na busca por este nó são visitados todos os nós ativos e, durante este procedimento, eliminam-se aqueles nós, cujos limitantes inferiores locais sejam menores do que o valor da melhor solução viável disponível no momento.

Estratégias de ramificação

Este é um dos pontos mais críticos de um algoritmo *branch-and-price*. Ao fazer o processo tradicional de ramificação sobre as variáveis do problema mestre, criam-se algumas dificuldades para o algoritmo.

A primeira delas, consiste no fato de que os problemas auxiliares seriam

desbalanceados. Entenda-se por desbalanceado o fato de estarmos ou não durante o processo de ramificação restringindo o espaço de soluções dos problemas auxiliares. Por exemplo, o espaço de soluções do problema auxiliar ao fixarmos o valor de y_l em um ficaria muito restritivo. Já no espaço de soluções para o problema auxiliar quando fixarmos y_l em zero resultaria em apenas uma coluna sendo excluída, e assim fazendo com este problema auxiliar não seja muito mais restritivo do que o problema auxiliar do seu nó predecessor.

A outra dificuldade reside nas estratégias de ramificação tradicionais impõem restrições adicionais ao problema auxiliar que destroem a sua estrutura.

Para superar as dificuldades apresentadas acima, os algoritmos *branch-and-price* fazem uso de uma regra definida por Ryan e Foster [63]. Esta regra é baseada no princípio de *set partitioning* e está fundamentada na seguinte proposição:

Proposição 5.1 *Se A é uma matriz 0-1 e uma solução básica para $Ay = 1$ é fracionária, isto é, no mínimo um dos componentes de y é fracionário, então existem duas restrições r e s do problema mestre, tal que*

$$0 < \sum_{k:a_{rk}=a_{sk}=1} y_k < 1. \tag{5.38}$$

Através do resultado anterior, podemos gerar o seguinte par de restrições para a fase de ramificação:

$$\sum_{k:a_{rk}=a_{sk}=1} y_k = 0 \tag{5.39}$$

e

$$\sum_{k:a_{rk}=a_{sk}=1} y_k = 1. \tag{5.40}$$

No nosso caso, essa regra implicaria a geraçãode dois problemas auxiliares; no primeiro deles, as localidades r e s devem obrigariamente ser cobertas por colunas diferentes (ramo da esquerda), isto é, por agrupamentos diferentes; e

no segundo caso, as localidades r e s devem ser cobertas pela mesma coluna (ramo da direita), portanto, pelo mesmo agrupamento.

Observe que pela Proposição 5.1, se não conseguirmos identificar um par de restrições para a fase de ramificação, então a solução para o problema mestre corresponde a uma solução inteira.

Procedimentos de *bounding*

O procedimento de *bounding* retorna um limitante inferior global que corresponde ao menor valor dentre os limitantes inferiores dos nós ativos.

Caso a árvore *branch-and-bound* tenha apenas o nó raiz, o valor do limitante inferior global é dado pelo valor da relaxação linear corrente. Caso este valor seja igual ao limitante inferior fornecido por uma solução viável conhecida, o problema estará resolvido.

Por outro lado, se tivermos pelo menos dois nós ativos, o valor do limitante inferior global corresponde ao menor valor da relaxação linear dentre todos os nós ativos.

5.5.3 Limitantes superiores

Como mencionado no Capítulo 4, o cálculo do limitante superior pode ser feito de duas formas. A primeira, que obtém o limitante superior inicial, faz uso da metaheurística *GRASP*. A segunda forma, calcula o limitante superior, tendo por base informações fornecidas pela relaxação linear do problema mestre.

Na próxima seção, detalharemos esta segunda forma de calcular (ou atualizar) os limitantes superiores.

5.5.4 Heurísticas primais

Até o final desta seção, descreveremos a heurística primal implementada para o problema de atribuição de localidades a anéis.

A heurística primal proposta baseia-se na escolha de alguns agrupamentos. Essa escolha é feita de acordo com o valor das variáveis, correspondentes a esses agrupamentos, na relaxação linear do problema mestre.

A princípio são escolhidos os agrupamentos cujo valor na relaxação seja maior ou igual a 0.5 e que satisfaçam a restrição de capacidade do anel federal. Com isso, obtemos uma solução viável. Em seguida, calcula-se o valor da função objetivo para estes agrupamentos e verifica-se se houve alguma melhoria no limitante superior.

Caso não seja possível encontrar uma solução viável, a heurística primal falha e o limitante inferior não pode ser atualizado.

5.5.5 Detecção de *tailing off*

Durante o processo de geração de colunas em um nó da árvore *branch-and-bound*, pode ocorrer que o valor da relaxação linear não sofra grandes alterações, ou seja, não melhore o valor da relaxação linear do problema mestre. A geração sucessiva de colunas, sem que esta melhoria se verifique, é o que caracteriza o fenômeno denominado de *tailing off*. Ao identificar a ocorrência deste fenômeno, sugere-se que a geração de colunas seja interrompida, realizando-se, então, uma ramificação.

No algoritmo implementado, a detecção de *tailing off* foi medida da seguinte forma. A fim de identificarmos se este processo está presente em um nó da árvore *branch-and-bound*, estabelecemos que se após *lnlp* resoluções consecutivas do problema mestre não tivermos obtido um ganho de pelo menos $p\%$ no valor da relaxação linear, com o acréscimo das colunas geradas, então faremos uma ramificação, interrompendo o processo de geração de colunas.

Para o algoritmo *branch-and-price* que implementamos, foi utilizado o valor 50 para o parâmetro *lnlp*, enquanto que para o parâmetro p, o valor estipulado foi igual a 10^{-1}.

5.5.6 Estratégias de seleção de colunas

O processo de geração de colunas é uma das partes mais importantes de um algoritmo *branch-and-price*. Precisamos decidir se a cada iteração do algoritmo *branch-and-price*, faremos uso de um método exato ou de uma heurística para a obtenção das colunas e, ainda, se adicionaremos uma única coluna ou múltiplas colunas.

A princípio, faremos uma comparação entre os resultados obtidos com o emprego de algoritmos exatos ou heurísticas na resolução do problema auxiliar. Através destes resultados, definiremos a quantidade de colunas a serem adicionadas ao problema mestre.

Forma heurística *versus* exata

Dado que inicializamos o algoritmo *branch-and-price* com um conjunto inicial de colunas (solução viável para o problema mestre), precisamos verificar se existe alguma variável que não está na formulação atual e que possua custo reduzido maior do que um, ou seja, precisamos resolver o problema auxiliar. Caso exista uma variável não básica com custo reduzido maior do que um, salvo degenerecência, a introdução desta variável na base do problema mestre melhora a solução atual. Caso contrário, a base atual corresponde a uma solução ótima.

Para o problema de atribuição de localidades a anéis, a verificação mencionada acima será feita através de métodos exatos e de uma heurística. O método exato é resolvido com o emprego do resolvedor linear XPRESS, aplicado à formulação (P5).

Já pelo aspecto heurístico, fizemos uso de algoritmo muito parecido com o Algoritmo 5, definido para o processo de geração das colunas iniciais. A única diferença é que, após verificar a viabilidade do agrupamento, devemos calcular o valor da função objetivo (5.17), com base nos custos duais (μ_i,π). E claro, não fazemos uso do passo 1.

Tanto para o algoritmo exato quanto para a heurística, estabelecemos um número máximo de colunas que podem ser geradas. No caso da heurística este valor corresponde a $n/4$, enquanto que para o método exato só geramos uma única coluna.

As colunas que porventura venham a ser geradas são armazenadas gradativamente em uma estrutura de dados, para serem adicionadas, em seguida, ao problema mestre do nó corrente. Todas as colunas pertencentes ao PL deste problema mestre são vistas apenas pelos descendentes deste nó na árvore *branch-and-bound.*

Adotamos o seguinte critério para avaliar a qualidade dos métodos empregados na geração de colunas: o número de PLs resolvidos e o tempo computacional obtido para a resolução de algumas instâncias do SRAP. Essas instâncias são as mesmas descritas no Capítulo 4.

Definidos os critérios de avaliação, passamos a comparar a implementação dos métodos empregados na geração de colunas para o problema mestre. Vale ressaltar que, como precisamos provar a otimalidade da base atual, sempre que não for possível encontrar uma coluna de forma heurística, geramos uma única coluna através do método exato.

			# Colunas					
Nome	# Nds	#PLs	Ini	Heur	Ext	z_{ub}	z_{lb}	Tmp
gl_15_1	1	35	76	24	25	3	3	0.94
gl_15_7	1	76	76	159	22	3	3	1.10
gl_15_9	1	30	76	26	20	3	3	0.72
gl_25_1	1	120	126	289	70	4	4	7.62
gl_25_3	1	153	126	360	90	3	3	5.15
gl_25_7	1	99	126	102	76	3	3	4.40
gl_25_8	1	204	126	665	87	4	4	7.83
gl_30_1	1	475	151	2255	129	4	4	19.34
gl_30_2	1	265	151	652	165	4	4	16.21
gl_30_3	1	214	151	600	123	4	4	15.36
gl_30_4	1	229	151	560	141	4	4	11.64
gh_25_7	1	288	126	1329	57	4	4	11.42
gh_25_8	1	132	126	170	99	3	3	9.95
gh_25_9	1	153	126	291	98	3	3	9.14
rl_30_4	1	473	151	2444	111	4	4	15.67
rl_30_5	1	204	151	511	128	4	4	11.72
rl_30_10	1	255	151	991	105	4	4	11.57
rh_25_1	1	255	126	937	87	3	3	13.69
rh_25_2	1	127	126	154	99	3	3	13.64
rh_25_3	1	140	126	172	106	3	3	12.94
rh_25_4	1	181	126	622	73	3	3	22.57
rh_25_10	1	102	126	58	90	3	3	7.06
rh_30_3	1	153	151	206	120	3	3	9.54
rh_30_4	1	200	151	349	144	3	3	25.27
rh_30_8	1	153	151	258	111	3	3	14.53

Tabela 5.4: Resultados computacionais obtidos com a resolução do problema auxiliar por heurísticas.

Foram testadas 25 instâncias da classe C1. Os resultados obtidos para estas instâncias são apresentados nas Tabelas 5.4 e 5.5. Na Tabela 5.4 temos os resultados obtidos com a heurística, enquanto que na Tabela 5.5 temos os resultados obtidos apenas com o emprego do método exato.

A coluna `Nome` indica a instância testada, a coluna `# Nds` indica o número de nós abertos na árvore *branch-and-bound*, a coluna `# PLs` indica o número de problemas lineares que foram resolvidos, a coluna `Ini` indica o número de colunas iniciais, a coluna `Heur` indica o número de colunas obtidas de forma heurística, a coluna `Ext` indica o número de colunas obtidas de forma exata. Observe que estas duas colunas aparecem na tabela que representa os resultados obtidos com a heurística. Isto se deve ao fato de termos que provar a otimalidade da base atual, como dito anteriormente. Por último, as colunas z_{ub}, z_{lb} e `Tmp` indicam, respectivamente, o limite superior fornecido pelo *GRASP*, o limite inferior obtido pelo algoritmo *branch-and-price* e o tempo computacional gasto na resolução de uma instância do SRAP.

Comparando as Tabelas 5.4 e 5.5, observamos que a geração de colunas usando apenas o método exato foi a que apresentou um melhor desempenho, visto que esta estratégia seguiu o compromisso estabelecido entre os critérios definidos para esses experimentos, ou seja, foram resolvidos poucos PLs e em um tempo computacional razoável.

Além do fator mencionado acima, nota-se que com o emprego do método exato foram geradas bem menos colunas do que com a heurística. Esse número de colunas chegou a ser 7 vezes (`gh_25_7`), 8 vezes (`gl_30_1`) e até 9 vezes (`rl_30_4`) menor, quando empregamos o método. Esta vantagem, com certeza, poderá se refletir diretamente no desempenho do algoritmo *branch-and-price*.

Devido aos resultados apresentados acima, optamos por utilizar métodos exatos na implementação do processo de geração de colunas. Apesar desta escolha, continuamos interessados em obter um tempo computacional ainda menor na resolução, de forma exata, do problema auxiliar, a cada iteração do algoritmo *branch-and-price*. Esta preocupação deve-se ao fato de que ainda gastamos uma quantidade razoável de tempo na resolução daquele problema.

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_{lb}	Tmp
			Ini	Ext			
gl_15_1	1	22	76	21	3	3	0.72
gl_15_7	1	34	76	33	3	3	1.24
gl_15_9	1	34	76	33	3	3	0.98
gl_25_1	1	68	126	67	4	4	7.01
gl_25_3	1	87	126	86	3	3	4.93
gl_25_7	1	102	126	101	3	3	4.76
gl_25_8	1	107	126	106	4	4	8.79
gl_30_1	1	132	151	131	4	4	14.46
gl_30_2	1	122	151	121	4	4	8.41
gl_30_3	1	128	151	127	4	4	13.19
gl_30_4	1	142	151	141	4	4	9.87
gh_25_7	1	90	126	89	4	4	13.22
gh_25_8	1	102	126	101	3	3	6.33
gh_25_9	1	119	126	118	3	3	12.03
rl_30_4	1	145	151	144	4	4	12.44
rl_30_5	1	134	151	133	4	4	10.73
rl_30_10	1	144	151	143	4	4	13.68
rh_25_1	1	85	126	84	3	3	12.34
rh_25_2	1	102	126	101	3	3	9.88
rh_25_3	1	102	126	101	3	3	10.29
rh_25_4	1	98	126	97	3	3	25.64
rh_25_10	1	92	126	91	3	3	8.39
rh_30_3	1	102	151	101	3	3	6.29
rh_30_4	1	146	151	145	3	3	23.58
rh_30_8	1	136	151	135	3	3	22.64

Tabela 5.5: Resultados computacionais obtidos com a resolução do problema auxiliar por métodos exatos.

A seguir, discutiremos como usar o método exato de forma mais eficiente.

Uma única coluna *versus* múltiplas colunas

Vanderbeck, em sua tese de doutorado (vide Vanderbeck [75]), relata que se o problema auxiliar pode ser resolvido eficientemente, então, à adição de uma única coluna é superior a adição de múltiplas colunas. Entretanto, se a resolução do problema auxiliar é computacionalmente cara, sugere o emprego de múltiplas colunas ao invés de uma única coluna.

Assim, precisamos verificar se os argumentos apresentados acima valem para o problema de atribuição de localidades a anéis. Para isto, foram realizados alguns experimentos computacionais com essas duas estratégias: uma única coluna *versus* múltiplas colunas.

Para a estratégia de múltiplas colunas, utilizamos a rotina `DefineUISEntry` da biblioteca XOSL para obter as soluções inteiras. Toda vez que o XPRESS encontrava uma solução inteira, a partir da resolução do problema auxiliar, chamávamos um procedimento cuja funcionalidade era pegar esta solução e, em seguida, armazená-la numa estrutura para posterior adição ao problema mestre.

Estabelecemos que o número máximo de colunas que podem ser geradas, a cada execução do método exato, é igual a 30.

Nestes experimentos, foram testadas 20 instâncias. Como na seção anterior, estas instâncias são as mesmas utilizadas nos experimentos computacionais do Capítulo 4. A Tabela 5.6 mostra os resultados obtidos com o emprego do método exato para uma única coluna, enquanto que a Tabela 5.7 mostra os resultados obtidos com múltiplas colunas. A descrição destas tabelas é dada logo abaixo.

A primeira coluna indica a instância testada; a segunda, número de nós abertos na árvore *branch-and-bound*; a terceira, o número de problemas lineares que foram resolvidos; a quarta e quinta colunas indicam, respectivamente, o número de colunas iniciais e o número de colunas obtidas de forma exata. Por último, a sexta, sétima e oitava colunas indicam, respectivamente, o limite superior fornecido pelo *GRASP*, o limite inferior obtido pelo algo-

ritmo *branch-and-price* e o tempo computacional gasto na resolução de uma instância do SRAP.

Pode-se verificar que a estratégia que melhor se adequa ao problema de atribuição de localidades a anéis é a estratégia que adota múltiplas colunas. Das 20 instâncias testadas da classe C1, a estratégia que adota uma única coluna teve que resolver mais PLs do que a estratégia com múltiplas colunas, em todas estas instâncias.

Esta diferença no número de PLs resolvidos talvez seja um dos fatores que tornam a estratégia de múltiplas colunas a mais rápida, e é claro, a mais adequada na resolução do problema auxiliar. Pode-se observar que em algumas instâncias, como por exemplo, `gl_25_8`, `rh_30_8`, `rh_30_9` e `rh_30_10`, o tempo computacional gasto pela estratégia de múltipla escolha chegou a ser aproximadamente três vezes melhor do que a estratégia que gera uma única coluna.

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_{lb}	Tmp
			Iniciais	Exatas			
gl_15_1	1	22	76	21	3	3	0.72
gl_15_4	1	26	76	25	3	3	0.72
gl_15_7	1	34	76	33	3	3	1.24
gl_15_9	1	34	76	33	3	3	0.98
gl_25_1	1	68	126	67	4	4	7.01
gl_25_3	1	87	126	86	3	3	4.93
gl_25_4	1	95	126	94	4	4	6.09
gl_25_7	1	102	126	101	3	3	4.76
gl_25_8	1	107	126	106	4	4	8.79
rl_50_1	1	255	251	254	5	5	27.33
rl_50_5	3	204	251	203	5	4	19.38
rl_50_6	1	204	251	203	4	4	17.54
rh_30_1	1	153	151	152	3	3	12.56
rh_30_2	1	139	151	138	4	4	40.89
rh_30_4	1	146	151	145	3	3	23.46
rh_30_6	1	141	151	140	4	4	33.31
rh_30_7	1	127	151	126	4	4	26.70
rh_30_8	1	136	151	135	3	3	22.58
rh_30_9	1	131	151	130	3	3	20.31
rh_30_10	1	102	151	101	4	4	13.68

Tabela 5.6: Resultados computacionais obtidos com a seleção de uma única coluna.

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_{lb}	Tmp
			Iniciais	Exatas			
gl_15_1	1	13	76	24	3	3	0.48
gl_15_4	1	21	76	33	3	3	0.62
gl_15_7	1	13	76	27	3	3	0.45
gl_15_9	1	18	76	35	3	3	0.63
gl_25_1	1	42	126	106	4	4	4.78
gl_25_3	1	54	126	127	3	3	3.33
gl_25_4	1	53	126	102	4	4	3.56
gl_25_7	1	51	126	128	3	3	3.49
gl_25_8	1	46	126	127	4	4	4.17
rl_50_1	1	153	251	519	5	5	21.02
rl_50_5	3	153	251	717	5	4	17.17
rl_50_6	1	153	251	453	4	4	15.75
rh_30_1	1	102	151	284	3	3	8.41
rh_30_2	1	61	151	195	4	4	23.29
rh_30_4	1	68	151	200	3	3	12.81
rh_30_6	1	68	151	192	4	4	19.32
rh_30_7	1	57	151	179	4	4	13.91
rh_30_8	1	56	151	191	3	3	9.95
rh_30_9	1	58	151	189	3	3	9.41
rh_30_10	1	54	151	197	4	4	10.81

Tabela 5.7: Resultados computacionais obtidos com a seleção de múltiplas colunas.

5.6 Resultados computacionais

Nesta seção, apresentaremos os resultados obtidos com o emprego do algoritmo *branch-and-price* proposto neste trabalho para resolução exata do problema de atribuição de localidades a anéis.

O algoritmo *branch-and-price* proposto foi implementado na linguagem `C`. Utilizamos o XPRESS como o resolvedor linear (e inteiro). Todos os experimentos computacionais foram realizados em um computador `AMD K6 450MHz`, com `256 MBytes` de memória `RAM`. Para compararmos o desempenho do algoritmo proposto, utilizamos as instâncias descritas no Capítulo 4.

O número inicial de colunas é igual a $5 \times n + 1$. Para os outros parâmetros, *lnlp* e *%p*, foram utilizados os seguintes valores 50 e 10^{-1}, respectivamente.

As instâncias apresentadas nas tabelas desta seção podem ou não estar marcadas com símbolos. O primeiro símbolo, denotado por "+", indica as instâncias onde não foi possível obtermos um limite superior inicial. Neste caso, para essas instâncias, adotamos como limite superior inicial um valor igual a $\theta \times z_{lb}$, onde z_{lb} pode ser expresso por

$$z_{lb} = \lceil D/B \rceil.$$

O valor estipulado para θ foi igual a 2, após alguns ajustes de parâmetros no algoritmo *branch-and-price*. Note que o valor de z_{lb} foi definido no Capítulo 4 e aparece sob a forma da equação (4.8).

O segundo símbolo, denotado por "*", indica as instâncias onde a solução obtida na relaxação linear corresponde à solução ótima inteira.

As tabelas apresentadas nesta seção possuem as seguintes colunas. A primeira coluna indica a instância testada. A segunda e terceira colunas indicam, respectivamente, o número de nós abertos na árvore *branch-and-bound* e o número de problemas lineares resolvidos. A quinta e sexta colunas indicam o número de colunas iniciais e o de colunas geradas, respectivamente. A sexta coluna indica o valor do limitante superior. A sétima coluna indica o valor da relaxação linear obtido pelo algoritmo *branch-and-price* no nó raiz da

árvore *branch-and-bound*. A oitava coluna indica o valor ótimo. Por último, a nona coluna indica o tempo computacional gasto para provar a otimalidade de uma solução.

5.6.1 Classe C1

Os resultados computacionais obtidos com as instâncias pertencentes a esta classe são apresentados nas Tabelas 5.8-5.11. As instâncias utilizadas nesta fase dos experimentos são as mesmas descritas nos Capítulos 2 e 4.

Instâncias geométricas

Nesta categoria de instâncias, foram testadas instâncias com valores de $B = 155Mbs$ e $B = 622Mbs$. Os resultados obtidos na resolução destas instâncias são apresentadas, respectivamente, nas Tabelas 5.8 e 5.9.

Foram resolvidas 50 instâncias pelo algoritmo *branch-and-price*. Em duas instâncias, temos que a solução obtida na relaxação linear corresponde à solução inteira. Nas demais instâncias, o valor da relaxação linear, quando arredondado para cima, já é o valor de uma solução inteira (lembre-se que os custos de todas as variáveis são inteiros).

Note, ainda, que o número de nós abertos, na maioria das instâncias testadas, foi igual a 1. No que se refere ao tempo computacional, podemos dizer que o tempo obtido pelo algoritmo *branch-and-price* é muito melhor do que o obtido pela formulação proposta por Goldschmidt *et al.* [26]. Por exemplo, para as instâncias gl_15_1, gl_15_4, gl_15_7 e gl_15_9, a melhoria no tempo computacional foi de 1362, 1072, 4177 e 2976 vezes, respectivamente. Além disso, para instâncias de médio porte ($n = 25$) e grande porte ($n = 30$ ou $n = 50$), enquanto o tempo obtido pela formulação proposta por Goldschmidt *et al.* [26] era superior a 18,000 segundos, o tempo obtido pelo algoritmo *branch-and-price* foi inferior a 1 minuto, chegando em alguns casos a ser 1682 vezes (gh_25_10) menor.

Observe, ainda, que com o emprego do algoritmo *branch-and-price* conseguimos resolver otimamente, em menos de 1 minuto, todas as instâncias desta seção. Além disso, este tempo computacional foi menor do que 15

segundos em 40 instâncias.

Instâncias aleatórias

Os experimentos desta categoria de instâncias são apresentados nas Tabelas 5.10 e 5.11. A primeira tabela indica os resultados obtidos para as instâncias com $B = 155Mbs$, enquanto que a segunda tabela indica os resultados com $B = 622Mbs$.

Durante os experimentos, testamos 61 instâncias. A exemplo das instâncias anteriores, temos em duas instâncias, `rh_15_8` e `rh_30_10`, que a solução obtida na relaxação linear corresponde à solução inteira. Mesmo não atingindo o ótimo inteiro ainda no primeiro nó da árvore *branch-and-bound*, a qualidade das colunas geradas continou boa, visto que nas demais instâncias o valor da relaxação linear, quando arredondado para cima, já é o valor de uma solução inteira.

Em apenas 6 das 61 instâncias foi necessário criar mais de 1 nó na árvore *branch-and-bound*. A exemplo das instâncias geométricas, o número máximo de nós aberto foi igual a 3. Mas, mesmo assim, o tempo computacional gasto não levou mais do que 30 segundos. Isto é muito bom, visto que para estas instâncias sabe-se que a formulação (P1) levou mais do que 18,000 segundos na resolução delas. Além disso, o valor obtido no primeiro nó, para estas instâncias, é bastante expressivo, quando comparado ao ótimo inteiro.

No que se refere ao tempo computacional, podemos dizer que ele continuou muito bom, e só excedemos o tempo de 1 minuto na instância `rh_50_1`. Mesmo assim, estes tempos computacionais são melhores do que os obtidos pela formulação proposta por Goldschmidt *et al.* [26]. Por exemplo, para as instâncias aleatórias com $B = 155Mbs$ ou $B = 622Mbs$, e n = 15, obtemos uma melhoria de, no mínimo, 5 vezes (`rl_15_3`) e, no máximo, 4425 vezes (`rh_15_9`), com relação ao tempo obtido pela formulação (P1). Este mesmo desempenho pode ser observado para as instâncias com n = 25, onde obtemos uma melhoria de, no mínimo, 15 vezes (`rh_25_9`) e de, no máximo, 1871 vezes (`rh_25_10`).

Observe que com o emprego do algoritmo *branch-and-price* foi possível

resolver, de maneira ótima, 46 instâncias, em menos de 15 segundos. A exemplo das instâncias geométricas, o tempo computacional aumentou, à medida que aumentamos o número de localidades. Porém, este tempo foi inferior a 1 minuto, com exceção da instância `rh_50_1`, e, assim, melhor do que o obtido pela formulação proposta por Goldschmidt *et al.* [26] que ultrapassava o tempo máximo estipulado, ou seja, 18,000 segundos.

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gl_15_1	1	13	76	24	3	2.83	3	0.48
gl_15_4	1	21	76	33	3	2.67	3	0.62
gl_15_7(*)	1	13	76	27	3	3	3	0.45
gl_15_9	1	18	76	35	3	2.22	3	0.63
gl_25_1	1	42	126	106	4	3.39	4	4.78
gl_25_3	1	54	126	127	3	2.93	3	3.33
gl_25_4	1	53	126	102	4	3.25	4	3.56
gl_25_7	1	51	126	128	3	2.56	3	3.49
gl_25_8	1	46	126	127	4	3.96	4	4.17
gl_30_1	1	71	151	195	4	3.56	4	6.67
gl_30_2	1	90	151	205	4	3.43	4	7.81
gl_30_3	1	84	151	198	4	3.25	4	8.96
gl_30_4	1	81	151	195	4	3.10	4	6.79
gl_30_5(*)	1	109	151	242	3	3	3	9.92
gl_30_6	1	77	151	182	4	3.13	4	7.09
gl_30_7	1	65	151	161	4	3.26	4	10.13
gl_30_8	1	68	151	167	4	3.44	4	8.96
gl_30_9	1	75	151	186	4	3.15	4	5.87
gl_50_1	3	153	251	440	6	4.61	5	24.29
gl_50_4	1	153	251	473	6	5.06	6	45.57
gl_50_7	1	136	251	414	5	4.42	5	46.29
gl_50_8	3	153	251	465	6	4.84	5	29.43
gl_50_9	1	153	251	373	4	3.57	4	15.04

Tabela 5.8: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C1 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gh_15_1	1	23	76	53	3	2.14	3	1.18
gh_15_2	1	20	76	33	3	2.33	3	1.72
gh_15_3	1	18	76	32	2	1.86	2	0.60
gh_15_6	1	26	76	53	2	1.75	2	0.77
gh_15_7	1	30	76	46	2	1.50	2	0.57
gh_15_8	1	21	76	58	3	2.85	3	2.14
gl_15_9	1	18	76	29	3	2.39	3	1.12
gh_15_10	1	31	76	55	2	1.59	2	0.73
gh_25_1	1	55	126	149	3	2.30	3	5.56
gh_25_2	3	77	126	194	3	1.95	2	4.05
gh_25_3	1	92	126	226	2	1.95	2	6.43
gh_25_4	1	63	126	136	3	2.71	3	7.53
gh_25_7	1	47	126	129	4	3.45	4	9.54
gh_25_8	1	61	126	156	3	2.56	3	6.89
gh_25_9	1	60	126	150	3	2.60	3	6.10
gh_30_1	1	87	151	229	3	2.61	3	8.58
gh_30_2	1	63	151	161	4	3.25	4	14.19
gh_30_3	1	49	151	147	4	3.40	4	9.42
gh_30_4	1	116	151	267	3	2.75	3	18.51
gh_30_5	1	51	151	143	4	3.30	4	10.01
gh_30_9	1	62	151	170	4	3.14	4	10.46
gh_30_10	1	102	151	270	3	2.19	3	8.26
gh_50_1	3	153	251	459	6	4.53	5	43.31
gh_50_4	3	153	251	458	5	3.94	4	35.04
gh_50_5	1	153	251	464	5	4.21	5	37.03
gh_50_6	1	153	251	459	5	4.71	5	44.85
gh_50_7	1	153	251	495	5	4.38	5	42.59

Tabela 5.9: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C1 ($B = 622Mbs$).

			# Colunas					
Nome	# Nds	#PLs	Iniciais	Exatas	z_{ub}	z_r	z_{lb}	Tmp
rl_15_1	1	21	76	42	3	2.36	3	0.84
rl_15_3	1	24	76	44	2	1.73	2	0.58
rl_15_4	1	23	76	41	3	2.85	3	1.39
rl_15_6	1	24	76	49	3	2.25	3	0.74
rl_15_8	1	22	76	39	3	2.50	3	1.01
rl_15_9	1	21	76	36	3	2.30	3	0.68
rl_15_10	1	20	76	40	3	2.95	3	0.99
rl_25_2	1	44	126	126	3	2.84	3	2.89
rl_25_4	1	38	126	108	4	3.28	4	4.10
rl_25_5	1	40	126	121	4	3.04	4	2.64
rl_25_6	1	47	126	121	4	3.31	4	5.09
rl_25_7	1	32	126	101	4	3.40	4	3.63
rl_25_8	1	47	126	121	3	2.57	3	2.59
rl_25_9	1	50	126	138	4	3.34	4	5.16
rl_25_10	1	44	126	130	4	3.29	4	4.22
rl_30_1	1	66	151	174	3	2.74	3	5.73
rl_30_4	1	69	151	180	4	3.79	4	7.34
rl_30_5	1	73	151	201	4	3.06	4	7.43
rl_30_7	1	70	151	198	3	2.92	3	7.69
rl_30_8	1	62	151	190	4	3.17	4	10.48
rl_30_10	1	76	151	215	4	3.50	4	8.41
rl_50_1	1	153	251	519	5	4.22	5	21.02
rl_50_3	3	153	251	492	5	3.81	4	21.10
rl_50_4	3	153	251	515	5	3.83	4	25.72
rl_50_5	3	153	251	717	5	3.65	4	17.17
rl_50_6	1	153	251	453	4	3.62	4	15.75
rl_50_7	1	130	251	439	5	4.35	5	27.98
rl_50_10	3	153	251	481	5	3.68	4	22.81

Tabela 5.10: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C1 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
rh_15_1	1	22	76	49	3	2.37	3	1.86
rh_15_4	1	25	76	48	2	1.65	2	1.01
rh_15_5	1	23	76	49	3	2.23	3	1.57
rh_15_6	1	19	76	42	3	2.18	3	1.57
rh_15_7	1	27	76	49	2	1.74	2	1.20
rh_15_8(*)	1	26	76	47	2	2	2	1.67
rh_15_9	1	15	76	34	3	2.13	3	0.82
rh_25_1	1	44	126	132	3	2.95	3	6.71
rh_25_2	1	50	126	154	3	2.25	3	6.90
rh_25_3	1	53	126	147	3	2.27	3	6.37
rh_25_4	1	40	126	138	3	2.99	3	10.62
rh_25_5	1	45	126	148	4	3.29	4	11.11
rh_25_6	3	42	126	134	4	2.96	4	7.12
rh_25_7	1	70	126	185	3	2.23	3	11.52
rh_25_8	1	45	126	139	4	3.11	4	10.58
rh_25_9	1	89	126	204	2	1.58	2	4.94
rh_25_10	1	64	126	167	3	2.06	3	7.41
rh_30_1	1	102	151	284	3	2.09	3	8.41
rh_30_2	1	61	151	195	4	3.12	4	23.29
rh_30_3	1	88	151	262	3	2.17	3	9.34
rh_30_4	1	68	151	200	3	2.48	3	12.81
rh_30_6	1	68	151	192	4	3.38	4	19.32
rh_30_7	1	57	151	179	4	3.12	4	13.91
rh_30_8	1	56	151	191	3	2.89	3	9.95
rh_30_9	1	58	151	189	3	2.74	3	9.41
rh_30_10(*)	1	54	151	197	4	3	3	10.81
rh_50_1	1	153	251	559	4	3.54	4	69.65
rh_50_2	3	153	251	609	4	2.78	3	32.46
rh_50_5	1	153	251	468	3	2.63	3	25.46
rh_50_6	1	153	251	584	3	2.67	3	32.79
rh_50_8	1	153	251	576	4	3.43	4	41.05
rh_50_9	1	153	251	569	4	3.17	4	37.15
rh_50_10	1	134	251	476	5	3.89	4	57.94

Tabela 5.11: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C1 ($B = 622Mbs$).

5.6.2 Classe C2

Os experimentos desta classe são apresentados nas Tabelas 5.12-5.18. As instâncias utilizadas nesta fase dos experimentos são as mesmas descritas nos Capítulos 2 e 4.

Instâncias geométricas

Nesta categoria de instâncias, foram testadas instâncias com valores de $B = 155Mbs$ e $B = 622Mbs$. Os resultados obtidos na resolução destas instâncias, com o emprego do algoritmo *branch-and-price*, são apresentadas, respectivamente, nas Tabelas 5.12, 5.13, 5.14 e 5.15.

O algoritmo *branch-and-price* voltou a superar os resultados obtidos pelo procedimento *branch-and-bound*. Foram testadas 140 instâncias. Destas 140 instâncias, temos que a solução obtida na relaxação linear corresponde à solução inteira em 10 instâncias. Percebemos, ainda, que nas demais instâncias, o valor da relaxação linear, quando arredondado para cima, já é o valor de uma solução inteira.

Note que, de acordo com os resultados apresentados nas Tabelas 5.12-5.15, observamos que em 65 instâncias tivemos que abrir mais de 1 nó na árvore *branch-and-bound*. Este número de nós, para algumas instâncias, foi acima de 5 nós. Isto não vem chega a ser uma grande desvantagem, visto que na maioria destas instâncias os valores obtidos no primeiro nó são bastante expressivos, quando comparados ao ótimo inteiro.

No que se refere ao tempo computacional, podemos dizer que o algoritmo *branch-and-price* continua superando o obtido pela formulação (P1). Conseguimos resolver, de maneira ótima, 104 instâncias em menos de 1 minuto. Percebemos, ainda, que em 69 destas 104 instâncias, o tempo computacional obtido com o emprego do algoritmo *branch-and-price* foi inferior a 15 segundos.

Fica evidente que os tempos computacionais obtidos são melhores do que os obtidos pela formulação proposta por Goldschmidt *et al.* [26]. Em algumas instâncias é possível caracterizar essa superioridade, por exemplo, para a instância `new.gl_50_6.3`, a formulação (P1) levou mais de 18,000 segundo,

enquanto que o algoritmo *branch-and-price* gastou apenas 4 minutos. Inclusive este foi o maior tempo que o algoritmo *branch-and-price* levou para a resolução das instâncias pertencentes a esta categoria.

Instâncias aleatórias

As Tabelas 5.16, 5.17 e 5.18 mostram os resultados obtidos com o emprego do algoritmo *branch-and-price* na resolução das 90 instâncias pertencentes a esta categoria.

Em 17 das 90 instâncias, temos que a solução obtida na relaxação linear corresponde à solução inteira. Percebemos, novamente, que nas demais instâncias o valor da relaxação linear, quando arredondado para cima, corresponde ao valor de uma solução inteira.

Observe que o número máximo de nós abertos na árvore *branch-and-bound* foi igual a 5, sendo que em 62 das 90 instâncias testadas, só foi necessário criar o nó raiz desta árvore. Além disso, como mencionado acima, o valor obtido no primeiro nó da árvore *branch-and-bound* pelo algoritmo *branch-and-price*, para estas instâncias, é bastante expressivo quando comparado ao ótimo inteiro.

Em 89 das 90 instâncias, o tempo computacional obtido pelo algoritmo *branch-and-price* foi inferior a 1 minuto, sendo que em 63 destas 89 instâncias, o tempo computacional ficou abaixo de 15 segundos. Assim, percebemos que o tempo computacional continuou bom, visto que conseguimos resolver otimamente todas as instâncias testadas em menos de 4 minutos.

Nota-se que os tempos computacionais obtidos nestes experimentos continuaram sendo melhores do que os obtidos pela formulação proposta por Goldschmidt *et al.* [26]. Em algumas instâncias, por exemplo, `new.rl_15_5.2` e `new.rh_15_10.6`, temos que o tempo computacional obtido foi de 9802 e 8586 vezes, respectivamente, melhor do que os tempos computacionais obtidos pela formulação (P1). Além disso, o algoritmo *branch-and-price* provou a otimalidade destas instâncias em menos de 2 segundos.

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gl_15_3.1	1	14	76	33	4	3.28	4	0.41
gl_15_3.2	1	12	76	32	4	3.18	4	0.48
gl_15_3.3	1	13	76	30	4	3.25	4	0.39
gl_15_3.4	1	29	76	49	4	3.19	4	1.16
gl_15_3.5	1	15	76	27	4	3.20	4	0.45
gl_15_3.6	1	12	76	32	4	3.18	4	0.48
gl_15_3.7	1	29	76	49	4	3.19	4	1.18
gl_15_3.8	1	13	76	30	4	3.25	4	0.42
gl_15_3.9	1	15	76	27	4	3.20	4	0.45
gl_15_3.10	1	14	76	33	4	3.29	4	0.40
gl_15_6.1(*)	1	20	76	39	3	3	3	1.14
gl_15_6.2(*)	1	23	76	50	3	3	3	1.15
gl_15_6.3(*)	1	19	76	38	3	3	3	0.92
gl_15_6.4(*)	1	20	76	51	3	3	3	0.89
gl_15_6.5	1	21	76	42	3	2.98	3	0.90
gl_15_6.6	1	21	76	49	3	2.94	3	1.04
gl_15_6.7(*)	1	25	76	52	3	3	3	1.22
gl_15_6.8(*)	1	11	76	29	3	3	3	0.44
gl_15_6.9(*)	1	12	76	32	3	3	3	0.39
gl_15_6.10(*)	1	12	76	24	3	3	3	0.55
gl_25_9.1	1	40	126	120	4	3.61	4	2.11
gl_25_9.2	1	43	126	107	4	3.38	4	2.76
gl_25_9.3	1	54	126	154	4	3.42	4	4.24
gl_25_9.4	1	28	126	92	4	3.42	4	1.82
gl_25_9.5	1	45	126	116	4	3.33	4	3.32
gl_25_9.6	1	35	126	120	4	3.36	4	2.73
gl_25_9.7	1	42	126	130	4	3.97	4	1.79
gl_25_9.8	1	48	126	133	4	3.37	4	3.04
gl_25_9.9	1	49	126	133	4	3.33	4	3.14
gl_25_9.10	1	40	126	115	4	3.33	4	2.25
gl_25_10.1	1	60	126	118	5	4.29	5	6.39
gl_25_10.2	1	63	126	126	5	4.28	5	5.64
gl_25_10.3	1	65	126	141	5	4.31	5	6.72
gl_25_10.4	1	63	126	126	5	4.29	5	5.61
gl_25_10.5	1	59	126	144	5	4.32	5	5.22
gl_25_10.6	1	39	126	90	5	4.27	5	3.51
gl_25_10.7	1	52	126	119	5	4.90	5	2.85
gl_25_10.8	1	46	126	96	5	4.29	5	3.57
gl_25_10.9	1	62	126	143	5	4.29	5	5.36
gl_25_10.10	1	65	126	141	5	4.31	5	6.64

Tabela 5.12: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gl_30_10.1	1	53	151	129	5	4.74	5	5.06
gl_30_10.2	3	82	151	217	6	4.59	5	7.22
gl_30_10.3	1	69	151	178	5	4.49	5	7.39
gl_30_10.4	1	68	151	159	5	4.61	5	3.99
gl_30_10.5	3	68	151	189	6	4.47	5	6.80
gl_30_10.6	1	59	151	172	5	4.49	5	6.02
gl_30_10.7	1	60	151	153	5	4.50	5	4.63
gl_30_10.8	1	81	151	200	5	4.48	5	10.53
gl_30_10.9	1	80	151	194	5	4.48	5	15.04
gl_30_10.10	1	55	151	137	5	4.56	5	4.31
gl_50_6.1	11	284	251	769	8	5.69	6	67.35
gl_50_6.2(+)	3	180	251	571	10	5.89	6	24.56
gl_50_6.3	27	489	251	1307	7	5.59	6	235.51
gl_50_6.4(+)	3	271	251	818	10	5.61	6	58.59
gl_50_6.5(+)	3	190	251	545	10	5.69	6	26.53
gl_50_6.6	3	236	251	691	7	5.49	6	179.49
gl_50_6.7(+)	3	203	251	618	10	5.69	6	34.46
gl_50_6.8(+)	3	215	251	597	10	5.60	6	89.20
gl_50_6.9(+)	3	204	251	623	10	5.50	6	52.97
gl_50_6.10(+)	3	194	251	563	10	5.54	6	53.67
gl_50_10.1	3	188	251	561	6	4.99	5	31.87
gl_50_10.2(*)	1	313	251	919	6	5	5	50.69
gl_50_10.3	3	355	251	968	7	5.21	6	30.93
gl_50_10.4	3	309	251	846	6	4.99	5	51.88
gl_50_10.5	3	204	251	599	7	4.98	5	31.69
gl_50_10.6(*)	1	271	251	690	7	5	5	36.26
gl_50_10.7(+)	3	280	251	801	8	5.09	6	31.04
gl_50_10.8	3	180	251	545	6	4.99	5	46.66
gl_50_10.9	3	321	251	878	7	5.06	6	63.54
gl_50_10.10(+)	3	214	251	642	8	5.25	6	20.73

Tabela 5.13: (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gh_25_5.1	1	50	126	127	4	3.42	4	7.63
gh_25_5.2	1	61	126	206	4	3.47	4	8.14
gh_25_5.3	1	55	126	158	4	3.39	4	7.61
gh_25_5.4	1	54	126	177	4	3.43	4	6.89
gh_25_5.5	1	42	126	148	4	3.44	4	6.56
gh_25_5.6	1	38	126	99	4	3.36	4	4.78
gh_25_5.7	1	63	126	175	4	3.38	4	8.61
gh_25_5.8	1	52	126	156	4	3.40	4	7.32
gh_25_5.9	1	46	126	143	4	3.47	4	4.55
gh_25_5.10	1	50	126	160	4	3.39	4	9.13
gh_30_8.1	1	86	151	233	4	3.61	4	8.02
gh_30_8.2	3	76	151	240	5	3.54	4	7.80
gh_30_8.3	1	81	151	264	4	3.58	4	7.55
gh_30_8.4	1	68	151	240	4	3.55	4	10.61
gh_30_8.5	1	82	151	254	4	3.57	4	9.38
gh_30_8.6	1	61	151	199	4	3.52	4	8.77
gh_30_8.7	1	52	151	188	4	3.54	4	8.10
gh_30_8.8	1	72	151	249	4	3.48	4	9.66
gh_30_8.9	1	81	151	254	4	3.50	4	12.05
gh_30_8.10	1	64	151	216	4	3.43	4	11.99
gh_50_2.1(+)	3	306	251	879	10	5.43	6	75.92
gh_50_2.2	1	245	251	748	6	5.53	6	64.61
gh_50_2.3(+)	5	265	251	812	10	5.49	6	97.96
gh_50_2.4(+)	3	355	251	1008	10	5.38	6	96.88
gh_50_2.5(+)	3	356	251	1035	10	5.39	6	78.69
gh_50_2.6	3	299	251	838	8	5.67	6	56.67
gh_50_2.7(+)	3	312	251	875	10	5.55	6	65.84
gh_50_2.8	3	357	251	1025	7	5.40	6	78.06
gh_50_2.9	1	255	251	771	6	5.37	6	104.11
gh_50_2.10(+)	3	379	251	1061	10	5.43	6	147.40
gh_50_3.1	5	139	251	511	6	4.85	5	87.16
gh_50_3.2	3	207	251	774	8	4.88	5	63.66
gh_50_3.3	1	160	251	559	5	4.83	5	73.31
gh_50_3.4	3	122	251	473	6	4.92	5	41.67
gh_50_3.5	3	132	251	489	6	4.91	5	37.88
gh_50_3.6(+)	3	145	251	588	8	4.91	5	43.52
gh_50_3.7(+)	3	210	251	810	8	4.96	5	61.17
gh_50_3.8	3	104	251	399	6	4.82	5	58.31
gh_50_3.9	3	204	251	770	6	4.78	5	69.74
gh_50_3.10(+)	11	211	251	766	8	5.48	6	95.96

Tabela 5.14: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 622Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
gh_50_8.1(+)	3	330	251	970	10	5.91	6	61.48
gh_50_8.2(+)	3	448	251	1275	10	5.71	6	71.14
gh_50_8.3	1	309	251	890	6	5.46	6	56.61
gh_50_8.4	1	277	251	890	6	5.42	6	56.57
gh_50_8.5(+)	3	302	251	951	10	5.96	6	58.38
gh_50_8.6(+)	3	340	251	977	10	5.63	6	61.88
gh_50_8.7(+)	3	314	251	1033	10	5.93	6	62.87
gh_50_8.8(+)	3	372	251	1146	10	5.90	6	63.65
gh_50_8.9(+)	19	486	251	1428	10	5.98	6	171.15
gh_50_8.10(+)	3	267	251	820	10	5.96	6	58.27
gh_50_9.1	7	221	251	753	7	4.70	5	100.27
gh_50_9.2	3	204	251	678	6	4.63	5	53.37
gh_50_9.3	3	172	251	566	6	4.69	5	70.03
gh_50_9.4(+)	3	196	251	702	8	4.95	5	46.93
gh_50_9.5(+)	3	166	251	567	8	4.72	5	58.23
gh_50_9.6(+)	3	232	251	894	8	4.75	5	69.20
gh_50_9.7(+)	3	190	251	679	8	4.95	5	49.39
gh_50_9.8(+)	1	215	251	685	6	4.66	5	67.53
gh_50_9.9(+)	1	160	251	521	5	4.69	5	136.99
gh_50_9.10	3	152	251	517	6	4.74	5	47.14
gh_50_10.1(+)	3	182	251	638	8	4.78	5	57.09
gh_50_10.2(+)	3	208	251	751	8	4.90	5	47.38
gh_50_10.3(+)	3	181	251	618	8	4.80	5	62.04
gh_50_10.4(+)	3	204	251	726	8	4.77	5	69.53
gh_50_10.5(+)	3	204	251	747	8	4.88	5	35.62
gh_50_10.6(+)	3	171	251	593	8	4.79	5	48.34
gh_50_10.7(+)	3	198	251	711	8	4.93	5	53.19
gh_50_10.8(+)	3	210	251	688	8	4.85	5	72.16
gh_50_10.9(+)	3	277	251	941	8	4.93	5	64.30
gh_50_10.10(+)	3	199	251	737	8	4.83	5	63.29

Tabela 5.15: (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias geométricas da classe C2 ($B = 622Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
rl_15_2.1(*)	1	19	76	38	3	3	3	0.69
rl_15_2.2	1	19	76	50	4	3.13	4	0.65
rl_15_2.3(*)	1	21	76	44	3	3	3	1.09
rl_15_2.4(*)	1	23	76	39	3	3	3	1.05
rl_15_2.5(*)	1	19	76	30	3	3	3	0.79
rl_15_2.6(*)	1	14	76	34	3	3	3	0.59
rl_15_2.7	5	19	76	39	4	2.99	3	0.77
rl_15_2.8	1	19	76	38	3	2.96	3	0.81
rl_15_2.9	1	15	76	33	4	3.06	4	0.43
rl_15_2.10(*)	1	17	76	35	3	3	3	0.70
rl_15_5.1	1	13	76	32	3	2.96	3	0.63
rl_15_5.2(*)	1	11	76	25	3	3	3	0.49
rl_15_5.3	1	15	76	37	3	2.95	3	0.64
rl_15_5.4(*)	1	11	76	25	3	3	3	0.49
rl_15_5.5(*)	1	19	76	43	3	3	3	0.72
rl_15_5.6(*)	1	11	76	25	3	3	3	0.48
rl_15_5.7	1	13	76	32	3	2.95	3	0.63
rl_15_5.8(*)	1	11	76	25	3	3	3	0.48
rl_15_5.9	1	13	76	32	3	2.96	3	0.62
rl_15_5.10(*)	1	11	76	25	3	3	3	0.51
rl_25_3.1	1	53	126	146	4	3.49	4	5.33
rl_25_3.2	1	53	126	126	4	3.53	4	5.09
rl_25_3.3	1	47	126	129	4	3.50	4	3.15
rl_25_3.4	1	49	126	132	4	3.56	4	4.64
rl_25_3.5	1	49	126	134	4	3.50	4	5.01
rl_25_3.6	1	45	126	137	4	3.61	4	4.86
rl_25_3.7	1	45	126	119	4	3.50	4	3.72
rl_25_3.8	1	45	126	122	4	3.50	4	3.75
rl_25_3.9	3	44	126	120	5	3.58	4	4.59
rl_25_3.10	1	45	126	117	4	3.50	4	4.19

Tabela 5.16: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
rl_30_2.1	3	73	151	180	5	3.86	4	8.62
rl_30_2.2	1	66	151	179	4	3.89	4	7.28
rl_30_2.3(+)	3	61	151	182	6	3.92	4	7.04
rl_30_2.4	1	60	151	174	4	3.95	4	5.22
rl_30_2.5	1	58	151	168	4	3.92	4	5.25
rl_30_2.6	3	68	151	183	5	3.81	4	12.11
rl_30_2.7	1	65	151	183	4	3.99	4	8.59
rl_30_2.8	1	61	151	177	4	3.96	4	5.59
rl_30_2.9	1	60	151	204	4	3.92	4	6.37
rl_30_2.10	3	86	151	216	5	3.91	4	14.83
rl_30_9.1	1	51	151	166	4	3.82	4	4.33
rl_30_9.2	1	66	151	189	4	3.83	4	4.94
rl_30_9.3	1	53	151	166	4	3.86	4	4.44
rl_30_9.4(+)	3	62	151	179	6	3.90	4	3.59
rl_30_9.5	1	56	151	146	4	3.79	4	5.46
rl_30_9.6	1	60	151	165	4	3.85	4	4.28
rl_30_9.7	1	65	151	185	4	3.90	4	3.36
rl_30_9.8	1	62	151	180	4	3.85	4	4.28
rl_30_9.9	1	53	151	147	4	3.81	4	4.06
rl_30_9.10(+)	3	63	151	177	6	3.85	4	4.83
rl_50_2.1(+)	3	196	251	586	8	4.79	5	33.03
rl_50_2.2(+)	3	173	251	513	8	4.71	5	199.39
rl_50_2.3(+)	3	164	251	520	8	4.93	5	20.71
rl_50_2.4(+)	3	201	251	616	8	4.89	5	25.86
rl_50_2.5(+)	3	172	251	555	8	4.83	5	19.85
rl_50_2.6(+)	3	162	251	513	8	4.72	5	26.31
rl_50_2.7(+)	3	177	251	533	8	4.81	5	19.74
rl_50_2.8(+)	3	190	251	572	8	4.87	5	25.91
rl_50_2.9(+)	5	193	251	627	8	4.87	5	32.95
rl_50_2.10(+)	3	193	251	632	8	4.82	5	27.12
rl_50_8.1(+)	3	174	251	552	8	4.98	5	25.85
rl_50_8.2(+)	3	151	251	481	8	4.91	5	20.86
rl_50_8.3(+)	3	173	251	632	8	4.97	5	29.94
rl_50_8.4(+)	5	191	251	842	8	4.99	5	36.46
rl_50_8.5(+)	3	160	251	481	8	5.06	6	18.46
rl_50_8.6(+)	3	206	251	682	8	4.89	5	29.26
rl_50_8.7(+)	3	178	251	514	8	4.97	5	23.39
rl_50_8.8(+)	3	172	251	583	8	4.96	5	24.38
rl_50_8.9(+)	3	169	251	563	8	4.99	5	33.33
rl_50_8.10(+)	3	189	251	584	8	4.95	5	39.94

Tabela 5.17: (cont.) Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 155Mbs$).

Nome	# Nds	#PLs	# Colunas		z_{ub}	z_r	z_{lb}	Tmp
			Iniciais	Exatas				
rh_15_10.1	1	26	76	44	3	2.87	3	2.69
rh_15_10.2	1	19	76	43	3	2.88	3	2.03
rh_15_10.3	1	25	76	50	3	2.99	3	3.40
rh_15_10.4(*)	1	15	76	49	3	3	3	2.33
rh_15_10.5	1	25	76	53	3	2.99	3	3.58
rh_15_10.6(*)	1	11	76	42	3	3	3	1.81
rh_15_10.7(*)	1	11	76	37	3	3	3	1.78
rh_15_10.8(*)	1	15	76	49	3	3	3	2.28
rh_15_10.9	1	16	76	54	3	2.97	3	1.72
rh_15_10.10(*)	1	10	76	37	3	3	3	1.72
rh_30_5.1	1	62	151	206	4	3.32	4	20.67
rh_30_5.2	1	65	151	210	4	3.35	4	19.30
rh_30_5.3	1	61	151	195	4	3.29	4	14.49
rh_30_5.4	1	58	151	183	4	3.36	4	19.36
rh_30_5.5	1	61	151	214	4	3.39	4	22.96
rh_30_5.6	1	58	151	195	4	3.35	4	19.71
rh_30_5.7	1	58	151	195	4	3.36	4	14.88
rh_30_5.8	1	50	151	192	4	3.38	4	20.44
rh_30_5.9	1	58	151	196	4	3.38	4	20.75
rh_30_5.10	1	49	151	186	4	3.37	4	11.54

Tabela 5.18: Resultados computacionais obtidos com o algoritmo *branch-and-price* para as instâncias aleatórias da classe C2 ($B = 622Mbs$).

5.7 Considerações

O desempenho computacional do algoritmo *branch-and-price* superou todos os tempos computacionais obtidos pela formulação proposta por Goldschmidt *et al.* [26]. Foram testadas 341 instâncias no total. Em todas elas foi possível provar a otimalidade em um tempo inferior a 4 minutos, chegando, às vezes a poucos segundos.

Vale ressaltar, também, que para algumas instâncias foi possível provar a otimalidade ainda no primeiro nó da árvore *branch-and-bound*, e com um tempo computacional muito inferior ao obtido pela formulação (P1). A Tabela 5.19 apresenta um resumo destas instâncias. A primeira coluna indica a instância testada, a segunda e a terceira indicam o tempo computacional obtido pela formulação (P1) e pelo algoritmo *branch-and-bound*, respectivamente, e por último, a quarta coluna indica quantas vezes o tempo obtido pelo algoritmo *branch-and-price* foi melhor do que o obtido pela formulação (P1). Os tempos computacionais estão expresssos em segundos. A presença do sinal "$>$" em algumas colunas indica que o valor presente naquela coluna corresponde ao valor mínimo.

Nota-se que o desempenho do algoritmo *branch-and-price* é, sem sombra de dúvida, melhor do que o da formulação (P1) na resolução, de forma exata, de instâncias do problema de atribuição de localidades a anéis.

Nome	Tempo	Tempo	# Vezes
gl_15_7	1880	0.45	4177
gl_30_5	>18000	9.92	>1814
rh_15_8	34	1.67	20
rh_30_10	>18000	10.81	>1665
gl_15_6.1	703	1.14	616
gl_15_6.2	633	1.15	550
gl_15_6.3	125	0.92	135
gl_15_6.4	577	0.89	648
gl_15_6.7	544	1.22	445
gl_15_6.8	925	0.44	2102
gl_15_6.9	666	0.39	1707
gl_15_6.10	407	0.55	740
gl_50_10.2	>18000	50.69	>355
gl_50_10.6	>18000	36.26	>496
rl_15_2.1	858	0.69	1243
rl_15_2.3	1221	1.09	1120
rl_15_2.4	380	1.05	361
rl_15_2.5	379	0.79	497
rl_15_2.6	250	0.59	423
rl_15_2.10	597	0.70	852
rl_15_5.2	4803	0.49	9802
rl_15_5.4	370	0.49	755
rl_15_5.5	558	0.72	775
rl_15_5.6	367	0.48	764
rl_15_5.8	366	0.48	762
rl_15_5.10	499	0.51	978
rh_15_10.4	9492	2.33	4073
rh_15_10.6	15542	1.81	8586
rh_15_10.7	6342	1.78	3562
rh_15_10.8	9333	2.28	4093
rh_15_10.10	30001	1.72	17442

Tabela 5.19: Comparação entre os desempenhos computacionais da formulação (P1) e do algoritmo *branch-and-price* na resolução de instâncias do SRAP.

Capítulo 6

Planos-de-corte geométricos para programação inteira 0-1

Neste capítulo desenvolvemos uma teoria que permite caracterizar uma família de planos-de-corte para problemas gerais de programacao linear inteira com variáveis binárias. Tais planos-de-corte possuem uma forte característica geométrica e, por isso, são denominados de *planos-de-corte geométricos*.

Nosso intuito com este estudo é utilizar estes planos-de-corte em um algoritmo *cut-and-branch* destinado a resolver o problema de construção dos anéis *SONET* idênticos de uma rede *backbone*.

O algoritmo *cut-and-branch* e os experimentos realizados com o mesmo são discutidos no Capítulo 7. Assim, daqui em diante, estaremos interessados na segunda fase da abordagem hierárquica do projeto de redes *backbone*, ou seja, na fase de construção dos anéis, a qual tem por entrada as localidades que compõem cada anel e que foram determinadas pela solução do SRAP.

Os planos-de-corte geométricos, a exemplo de outros introduzidos em Balas *et al.* [4], Boyd [10] e Wolsey [78, 79], são independentes da estrutura combinatória do problema a ser resolvido e, devido a esta propriedade, são algumas vezes referidos como planos-de-corte de propósito geral. Talvez o exemplo mais conhecido destes tipos de desigualdades são os planos-de-corte de Chvátal-Gomory [27].

Uma outra característica das desigualdades que definiremos neste capítulo é que, ao invés de eliminarmos soluções fracionárias, nos preocupamos com a eliminação de soluções inteiras inviáveis do problema de programação linear

inteira 0-1, associado a um dado problema combinatório.

Ainda neste capítulo, mostramos que existe uma correspondência de um para um entre os planos-de-corte geométricos e os planos-de-corte canônicos propostos por Balas e Jeroslow [5]. Mostramos, ainda, que as desigualdades de eliminação de subciclos para o problema do caixeiro viajante simétrico, denotado por PCVS, podem ser obtidas via planos-de-corte geométricos, e que outras desigualdades lineares, conhecidas para outros problemas, podem ser obtidas a partir dos planos-de-corte geométricos.

Em seguida, apresentaremos um teorema que possibilitará a compreensão de algumas provas deste capítulo.

6.1 Preliminares

Esta seção contém algumas definições e resultados que serão mencionados nas seções seguintes e que podem ser encontrados em qualquer livro de programação linear inteira (vide Nemhauser e Wolsey [59] e, Wolsey [80]). A única exceção é o Teorema 6.1, que propusemos e provamos no final desta seção.

Definição 6.1 *A* distância de Hamming *entre dois pontos x e y, denotada por $dist(x,y)$, corresponde ao número de componentes diferentes que possuem os pontos x e y, isto é, $dist(x,y) = \sum_{i=1}^{n} |x_i - y_i|$.*

Observa-se que a Definição 6.1 é válida para os vetores x e $y \in \{0,1\}$. Até o final deste capítulo, sempre que nos referirmos à distância entre pontos, estaremos nos referindo à distância de Hamming.

Definição 6.2 *Uma matriz $A_{m\times n}$ de inteiros é totalmente unimodular (TU) se o determinante de cada submatriz quadrada de A for igual a -1, 0 ou 1.*

Proposição 6.1 *Se A é TU então* $\begin{pmatrix} A \\ I \end{pmatrix}$ *é TU.*

Corolário 6.1 *Seja A uma matriz cujos elementos são iguais a $(-1,0,1)$, tal que cada coluna de A não possua mais do que dois elementos não nulos. Então, a matriz A é TU, se e somente se as linhas em A puderem ser particionadas em dois subconjuntos, Q_1 e Q_2, tal que*

1. *se ambos os elementos não nulos têm o mesmo sinal, então um desses elementos pertence a uma linha do subconjunto Q_1, e o outro elemento pertence a uma linha do subconjunto Q_2;*

2. *se os dois elementos não nulos têm sinais opostos, então ambos os elementos pertencem a linhas de um mesmo subconjunto.*

Definição 6.3 *Um poliedro $P \subseteq \mathbb{R}^n$ é* inteiro, *se todos os seus pontos extremos possuem coordenadas inteiras.*

Proposição 6.2 *Seja $P = \{x \in \mathbb{R}^n : Ax = b\}$ um poliedro não vazio. Suponha que A é uma matriz $m \times n$ e $b \in \mathbb{Z}^m$. Se A é TU, então P é inteiro.*

Definição 6.4 *Dado um poliedro $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ com dimensão cheia, diz-se que y é um ponto* interior *de P, se $Ay < b$.*

Proposição 6.3 *Todo poliedro com dimensão cheia possui um ponto interior.*

Proposição 6.4 *$S = \{x_0, x_1, \ldots, x_n\}$ é um conjunto de pontos afim independentes em $\mathbb{R}^n$, se e somente se os pontos em $S' = \{x_1 - x_0, x_2 - x_0, \ldots, x_n - x_0\}$ são linearmente independentes.*

Proposição 6.5 *Se $S = \{x_1, \ldots, x_n\}$ é um conjunto de pontos linearmente independentes em $\mathbb{R}^n$, então para cada constante $\alpha \neq 0$, os pontos em $S' = \{\alpha x_1, \ldots, \alpha x_n\}$ são também linearmente independentes.*

Em seguida, enunciamos o teorema que utilizaremos na prova de facetas deste capítulo.

Teorema 6.1 *Seja $P = \{x \in \mathbb{R}^n : \pi x \leq \pi_0, a_i x \leq b_i \text{ para } i = 1, \ldots, m\}$ um poliedro com dimensão cheia e $P' = \{x \in \mathbb{R}^n : a_i x \leq b_i \text{ para } i = 1, \ldots, m\}$. Se $P' \setminus P \neq \emptyset$, então $\pi x \leq \pi_0$ define uma faceta de P.*

Prova: Vamos iniciar a prova, assumindo que $\pi_0 \neq 0$, uma vez que se este é o caso, podemos sempre transladar a origem para satisfazer essa condição.

A partir da Proposição 6.3, existe um ponto interior y em P. Então, existe um escalar $\epsilon > 0$, tal que a bola com origem em y e raio igual a ϵ, denotada por $B(y, \epsilon)$, somente contém pontos que estão no interior de P. Além disso, já que $P' \setminus P \neq \emptyset$, existe um ponto $z \in P'$, tal que $\pi z > \pi_0$.

Considere o hiperplano passando através do ponto y e que é ortogonal a $(y - z)$. Este hiperplano é dado por $H = \{x \in \mathbb{R}^n : (y - z)^T x = (y - z)^T y\}$. Como $H \cap B(y, \epsilon)$ é uma esfera com dimensão igual a $(n - 1)$, com origem em y e com raio igual a ϵ, ela contém um conjunto afim independente de n pontos. Digamos que $\{y^1, \ldots, y^n\}$ sejam estes pontos. Agora, porque $z \notin H$, os pontos $\{z, y^1, \ldots, y^n\}$ são afim independentes. Assim, pela Proposição 6.4, os pontos em $\{y^1 - z, \ldots, y^n - z\}$ são linearmente independentes.

Dado que para cada $i = 1, \ldots, n$, o segmento $\overline{zy^i}$ intersecta o hiperplano $\{x \in \mathbb{R}^n : \pi x = \pi_0\}$, existe um escalar $\alpha_i > 0$, tal que $u^i = z + \alpha_i(y^i - z)$ satisfaz $\pi u^i = \pi_0$. Assim, u^i pertence a P e, a partir da Proposição 6.5, os pontos em $\{u^1 - z, \ldots, u^n - z\}$ são linearmente independentes. Agora, porque $\pi z \neq \pi_0$, de acordo com a Proposição 6.4, os pontos em $\{z, u^1, \ldots, u^n\}$ são afim independentes. Assim, a face $F(\pi, \pi_0) = \{x \in P : \pi x = \pi_0\}$ tem dimensão igual a $n - 1$. Portanto, a prova está concluída.

□

6.2 Planos-de-corte esféricos

Considere o seguinte problema de programação linear

(P6)

$$\begin{aligned} &\text{minimize } c^T x, \\ &\text{sujeito a:} \\ &\qquad Dx \leq b, \\ &\qquad x \in \mathbb{B}^n, \end{aligned}$$

onde $D \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, e $c \in \mathbb{R}^n$.

Assuma que o problema (P6) não corresponda a uma descrição minimal de um dado problema de otimização combinatória, ou seja, os pontos $x \in \mathbb{B}^n$ não correspondem a soluções viáveis para um dado problema em questão. Por exemplo, para o problema do caixeiro viajante simétrico, se assumirmos que $Dx \leq b$ representa apenas as desigualdades que estabelecem que o grau de cada vértice deve ser igual a 2, teríamos que toda solução obtida com a resolução de (P6) corresponderia a ciclos disjuntos.

O propósito desta seção consiste em definir uma desigualdade linear, denominada *plano-de-corte esférico*, que seja capaz de eliminar qualquer ponto $\overline{x} \in \mathbb{B}^n$, independente do sistema de desigualdades lineares $Dx \leq b$, isto é, pretendemos definir uma desigualdade linear que caracterize o conjunto de pontos $X_1(\overline{x}) = \mathbb{B}^n \setminus \{\overline{x}\}$.

Em seguida, descreveremos como é possível obter tal desigualdade linear através de noções de geometria.

6.2.1 Planos-de-corte 1-esférico

Considere uma esfera S com centro no ponto $\overline{x} \in \mathbb{B}^n$ e raio igual a 1. Assim, obtemos a seguinte equação:

$$\sum_{j=1}^{n}(x_j - \overline{x}_j)^2 = 1. \tag{6.1}$$

Como desejamos caracterizar o conjunto $X_1 = \mathbb{B}^n \setminus \{\overline{x}\}$, isto é, os pontos na superfície e fora da esfera S, então podemos reescrever a equação (6.1) como

$$\sum_{j=1}^{n}(x_j - \overline{x}_j)^2 \geq 1, \tag{6.2}$$

a fim de satisfazer essa caracterização.

Sabendo-se que $x_j^2 = x_j$, pois $x_j \in \{0, 1\}$ com $j = 1, \ldots, n$, podemos linearizar a desigualdade (6.2). Desta forma, obtemos

$$\sum_{j=1}^{n}(x_j^2 - 2x_j\overline{x}_j + \overline{x}_j^2) \geq 1 \Longrightarrow$$

$$\sum_{j=1}^{n}(x_j - 2x_j\overline{x}_j + \overline{x}_j) \geq 1 \Longrightarrow$$

$$\sum_{j=1}^{n}(1 - 2\overline{x}_j)x_j \geq 1 - \sum_{j=1}^{n}\overline{x}_j. \tag{6.3}$$

A desigualdade (6.3) é denominada *plano-de-corte 1-esférico* e foi inicialmente estudada em Maculan [50].

Note-se que a desigualdade (6.3) é válida para todos os pontos em $\mathbb{B}^n$, cuja distância de Hamming até o ponto $\overline{x}$ é, no mínimo, igual a 1. Note-se, ainda, que, os pontos 0-1 que satisfazem a desigualdade (6.3) na igualdade são precisamente aqueles pontos adjacentes ao ponto $\overline{x}$ no hipercubo unitário, com dimensão igual a n.

A Figura 6.1 apresenta um exemplo de um plano-de-corte 1-esférico. Considere o ponto $\overline{x} = (1, 1, 1)$. Logo, o plano-corte 1-esférico associado a este ponto pode ser expresso por $x_1 + x_2 + x_3 \leq 2$.

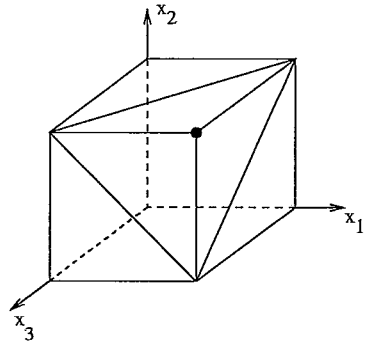

Figura 6.1: Plano-de-corte 1-esférico $x_1 + x_2 + x_3 \leq 2$ definido pelo ponto $\overline{x} = (1, 1, 1)$.

Assim, podemos definir o seguinte sistema linear:

$$S_1(\overline{x}) = \left\{ x \in \mathbb{R}^n : 0 \leq x_j \leq 1, \ j = 1, \ldots, n, \right.$$

$$\left. \sum_{j=1}^{n}(1 - 2\overline{x}_j)x_j \geq 1 - \sum_{j=1}^{n}\overline{x}_j \right\}. \tag{6.4}$$

Em seguida, mostramos que $S_1(\overline{x})$ é um poliedro inteiro e que a desigualdade (6.3) define uma faceta para $S_1(\overline{x})$.

Teorema 6.2 *O poliedro $S_1(\overline{x})$ é inteiro.*

Prova:

Considere um subsistema linear de $S_1(\overline{x})$, formado pelo plano-de-corte 1-esférico e pelas restrições de não negatividade das variáveis x. A matriz W de restrições deste subsistema é totalmente unimodular. Para compreender isto, basta observar que W não tem mais do que dois elementos não nulos em cada coluna, e que as linhas de W podem ser divididas em dois conjuntos $Q1$ e Q_2, de acordo com o Corolário 6.1, como se segue. A linha correspondente ao plano-de-corte 1-esférico é atribuída ao conjunto Q_1. Para cada $j \in \{1, \ldots, n\}$, a linha correspondente às restrições de não negatividade $x_j \geq 0$ é atribuída ao conjunto Q_1, se $\overline{x}_j = 1$; caso contrário, será atribuída ao conjunto Q_2.

Finalmente, a partir da Proposição 6.1, concluímos que o sistema linear $S_1(\overline{x})$ é totalmente unimodular e, portanto, o poliedro é inteiro.

□

Teorema 6.3 *O poliedro $S_1(\overline{x})$ possui dimensão cheia para cada ponto $\overline{x}$ em $\mathbb{B}^n$.*

Prova:

Basta considerar o caso onde $\overline{x} = (1, 1, \ldots, 1)$. Note-se que a prova para os outros pontos em $\mathbb{B}^n$ se segue a partir da rotação e translação. Assim, para $n > 1$, os pontos $(0, 0, \ldots, 0)$ e e_i, para $i = 1, \ldots, n$ pertencem a $X_1(\overline{x})$, onde e_i é um vetor de dimensão n, com todos os elementos iguais a 0, exceto na posição i, que é igual a 1. Portanto, X_1 tem $n+1$ pontos afim independentes. Assim, a prova esta concluída.

□

Teorema 6.4 *O plano-de-corte 1-esférico define uma faceta de $S_1(\overline{x})$ para cada ponto $\overline{x} \in \mathbb{B}^n$.*

Prova:

Novamente, a prova só será feita apenas para o ponto $\overline{x} = (0, 0, \ldots, 0)$. A prova é imediata a partir dos Teoremas 6.1 e 6.3.

□

6.2.2 Planos-de-corte p-esférico

Podemos generalizar os resultados apresentados na seção 6.2.1, da seguinte maneira. Dado um ponto $\overline{x} \in \mathbb{B}^n$, assuma que $W_p(\overline{x})$ seja o conjunto de todos os pontos em $\mathbb{B}^n$ que possuem distância de Hamming no máximo igual a $p-1$, com relação ao ponto $\overline{x}$, onde $0 < p < n$ e $n > 1$. Considere, ainda, que $X_p(\overline{x}) = \mathbb{B}^n \setminus W_p(\overline{x})$.

Os pontos pertencentes ao conjunto $X_p(\overline{x})$ satisfazem

$$\sum_{j=1}^{n}(x_j - \overline{x}_j)^2 \geq p. \tag{6.5}$$

Como na seção anterior, desde que $x_j^2 = x_j$, para todo $j = 1, \ldots, n$, podemos linearizar a desigualdade (6.5) e obter a seguinte desigualdade

$$\sum_{j=1}^{n}(1 - 2\overline{x}_j)x_j \geq p - \sum_{j=1}^{n}\overline{x}_j. \tag{6.6}$$

Esta desigualdade é denominada *plano-de-corte p-esférico.*

Como anteriormente, podemos definir o seguinte poliedro:

$$\mathrm{S}_p(\overline{x}) = \left\{ x \in \mathbb{R}^n : 0 \leq x_j \leq 1, \ j = 1, \ldots, n, \right.$$
$$\left. \sum_{j=1}^{n}(1 - 2\overline{x}_j)x_j \geq p - \sum_{j=1}^{n}\overline{x}_j \right\}. \tag{6.7}$$

Assumindo que $n \geq 2$ e $0 < p < n$, podemos mostrar resultados similares aos apresentados na seção 6.2.1.

Teorema 6.5 *O poliedro* $\mathrm{S}_p(\overline{x})$ *é inteiro.*

Prova:

A prova do Teorema 6.5 segue os mesmos passos do Teorema 6.2, apresentado na seção 6.2.1.

□

Teorema 6.6 *O poliedro* $\mathbb{S}_p(\overline{x})$ *possui dimensão cheia para todo ponto* $\overline{x}$ *em* $\mathbb{B}^n$.

Teorema 6.7 *O plano-de-corte p-esférico define uma faceta de* $\mathbb{S}_p(\overline{x})$ *para todo ponto* $\overline{x}$ *em* $\mathbb{B}^n$.

Para provar os Teoremas 6.6 e 6.7, primeiro assumimos que $\overline{x} = (1, 1, \ldots, 1)$. Então, basta observar que o conjunto de pontos representando as colunas da matriz abaixo pertencem a $\mathbb{S}_p(\overline{x})$ e que estão distantes de p, a partir do ponto $\overline{x}$. Claramente, os pontos definidos na matriz são afim independentes e já que o ponto $(0, 0, \ldots, 0)$ também pertence a $\mathbb{S}_p(\overline{x})$, temos todos os argumentos para completar as provas destes teoremas.

0	0	...	0	0	1	0	...	0	0	} $p-1$
0	0	...	0	0	0	1	...	0	0	
...	...	...	...	...	...	...	...	...	...	
0	0	...	0	0	1	0	...	1	0	
0	0	...	0	0	1	0	...	0	1	
0	1	...	1	1	0	1	...	1	1	} $n-p+1$
1	0	...	1	1	0	0	...	1	1	
1	1	...	1	1	1	0	...	1	1	
1	1	...	1	1	1	1	...	1	1	
...	...	...	...	...	...	...	...	...	...	
1	1	...	1	1	1	1	...	0	1	
1	1	...	0	1	1	1	...	0	0	
1	1	...	1	0	1	1	...	1	0	

Claramente, a desigualdade (6.6) generaliza a desigualdade (6.3), uma vez que esta última corresponde a um caso especial, onde p é igual a 1.

6.3 Planos-de-corte cilíndricos

Nesta seção, consideramos a possibilidade de definir uma desigualdade linear, denominada *plano-de-corte!cilíndrico*, que seja capaz de eliminar de $\mathbb{B}^n$ dois pontos $\overline{x}^1$ e $\overline{x}^2$, simultaneamente.

O exemplo na Figura 6.2 mostra que a adição dos dois planos-de-corte 1-esférico, definidos por $\overline{x}^1$ e $\overline{x}^2$, ao sistema linear que descreve o hipercubo unitário n-dimensional, não produz um poliedro inteiro. Além disso, é fácil verificar que se $\overline{x}^1$ e $\overline{x}^2$ não são adjacentes, não existe um hiperplano que separe $\overline{x}^1$ e $\overline{x}^2$, a partir dos pontos que restaram em $\mathbb{B}^n$.

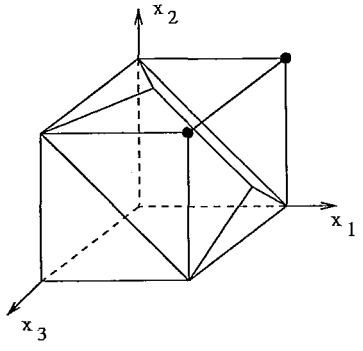

Figura 6.2: Plano-de-corte 1-esférico $x_1 + x_2 + x_3 \leq 2$ e $x_1 + x_2 - x_3 \leq 1$ definidos, respectivamente, pelos pontos $\overline{x} = (1, 1, 1)$ e $\overline{x}^2 = (1, 1, 0)$.

Assim, nosso objetivo é obter uma desigualdade linear, que seja capaz de eliminar um par de pontos adjacentes de $\mathbb{B}^n$, simultaneamente. Inicialmente, observamos que tais pontos diferem exatamente de uma componente, digamos $k \in \{1, \ldots, n\}$. Logo, para o par de pontos $(\overline{x}^1, \overline{x}^2)$, temos que a distância de Hamming é igual a 1, ou seja, $dist(\overline{x}^1, \overline{x}^2) = 1$.

6.3.1 Planos-de-corte 1-cilíndrico

Assuma que o ponto $\overline{x}^1$ difere do ponto $\overline{x}^2$ somente na k-ésima componente. Se desejamos separar estes pontos, então estamos interessados nos pontos de $\mathbb{B}^n$ que satisfazem

$$\sum_{\substack{j=1 \\ j \neq k}}^{n} (x_j - \overline{x}_j^1)^2 \geq 1. \tag{6.8}$$

Note-se que esta desigualdade mostra que o ponto x difere ou do ponto $\overline{x}^1$ ou do ponto $\overline{x}^2$, em no mínimo, uma componente. Em outras palavras, a distância de Hamming entre o ponto x e, por exemplo, o ponto $\overline{x}^1$ é, no mínimo, igual a 1.

Agora, vamos desenvolver o lado esquerdo da desigualdade acima. Sabendo-se que os pontos x e $\overline{x}_j^1$ são pontos 0–1, obtemos

$$\begin{aligned}
\sum_{\substack{j=1 \\ j\neq k}}^{n} [x_j^2 - 2x_j\overline{x}_j^1 + (\overline{x}_j^1)^2] &\geq 1 \Longrightarrow \\
\sum_{\substack{j=1 \\ j\neq k}}^{n} (x_j - 2x_j\overline{x}_j^1 + \overline{x}_j^1) &\geq 1 \Longrightarrow \\
\sum_{\substack{j=1 \\ j\neq k}}^{n} [(1 - 2\overline{x}_j^1)x_j + \overline{x}_j^1] &\geq 1 \Longrightarrow \\
\sum_{\substack{j=1 \\ j\neq k}}^{n} (1 - 2\overline{x}_j^1)x_j &\geq 1 - \sum_{\substack{j=1 \\ j\neq k}}^{n} \overline{x}_j^1. \qquad (6.9)
\end{aligned}$$

A desigualdade (6.9) é denominada de *plano-de-corte 1-cilíndrico*, com relação aos pontos $\overline{x}^1$ e $\overline{x}^2$.

A Figura 6.3 mostra um exemplo destas desigualdades para $\mathbb{B}^3$. Como se pode ver, a desigualdade é utilizada para eliminar os pontos de $\mathbb{B}^3$ que ficam no interior do cilindro, cujo eixo é definido pelos pontos $\overline{x}^1$ e $\overline{x}^2$.

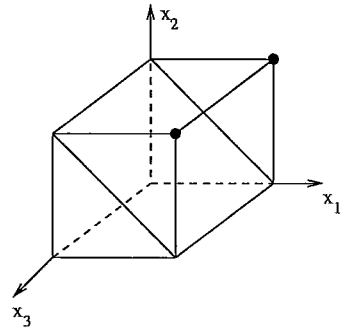

Figura 6.3: Plano-de-corte 1-cilíndrico $x_1 + x_2 \leq 1$ definido pelos pontos $\overline{x}^1 = (1, 1, 0)$ e $\overline{x}^2 = (1, 1, 1)$.

Na próxima seção, generalizamos os planos-de-corte 1-cilíndrico e realizamos um estudo sobre o poliedro obtido com a interseção do hipercubo unitário n-dimensional e o semi-espaço definido pela generalização da desigualdade (6.9).

6.3.2 Planos-de-corte p-cilíndrico

Suponha que $n > 2$ e $0 < p < n-1$. Assuma que $\overline{x}^1$ e $\overline{x}^2$ sejam dois pontos adjacentes em $\mathbb{B}^n$. Considere, ainda, que $W_p(\overline{x}^1, \overline{x}^2)$ representa o conjunto de pontos em $\mathbb{B}^n$, cuja distância de Hamming, com relação a $\overline{x}^1$ ou a $\overline{x}^2$, é estritamente menor do que p.

A fim de generalizar a desigualdade (6.9), definimos uma desigualdade linear válida para $X_p(\overline{x}^1, \overline{x}^2) = \mathbb{B}^n \setminus W_p(\overline{x}^1, \overline{x}^2)$. Se k é a única componente na qual $\overline{x}^1$ e $\overline{x}^2$ diferem, então esta desigualdade pode ser escrita como

$$\sum_{\substack{j=1 \\ j \neq k}}^{n} (1 - 2\overline{x}_j^1) x_j \geq p - \sum_{\substack{j=1 \\ j \neq k}}^{n} \overline{x}_j^1. \tag{6.10}$$

A desigualdade (6.10) é denominada de *plano-de-corte p-cilíndrico*, com relação aos pontos $\overline{x}^1$ e $\overline{x}^2$.

Vamos definir o poliedro $\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$ como sendo

$$\mathcal{C}_p(\overline{x}^1, \overline{x}^2) = \left\{ x \in \mathbb{R}^n : 0 \leq x_j \leq 1, \ j = 1, \ldots, n, \right.$$
$$\left. \sum_{\substack{j=1 \\ j \neq k}}^{n} (1 - 2\overline{x}_j^1) x_j \geq p - \sum_{\substack{j=1 \\ j \neq k}}^{n} \overline{x}_j^1 \right\}. \tag{6.11}$$

Então, o teorema seguinte mantém-se.

Teorema 6.8 *O poliedro $\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$ é inteiro.*

Prova:

Semelhante à prova do Teorema 6.2.

□

Teorema 6.9 *O poliedro $\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$ possui dimensão cheia para cada par de pontos, $\overline{x}^1$ e $\overline{x}^2$, adjacentes em $\mathbb{B}^n$.*

Prova:

A prova é feita, tomando-se os pontos $\overline{x}^1 = (1, 1, \ldots, 1, 1)$ e $\overline{x}^2 = (1, 1, \ldots, 1, 0)$, uma vez que os argumentos se mantêm válidos para qualquer par de pontos adjacentes em $\mathbb{B}^n$, aplicando a rotação e translação.

Note-se que, estamos supondo que $n > 2$ e $0 < p < n - 1$. Então, o ponto nulo e os pontos e_i, para todo $i = 1, \ldots, n$ pertencem a $\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$. Portanto, temos todos os pontos afim independentes para completar a prova.

□

Teorema 6.10 *O plano-de-corte p-cilíndrico define uma faceta de* $\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$ *para cada par de pontos adjacentes* $\overline{x}^1$ *e* $\overline{x}^2$ *em* $\mathbb{B}^n$.

Prova: Imediata, a partir dos Teoremas 6.1 e 6.9.

□

6.4 Forma geral dos planos-de-corte geométricos

Os planos-de-corte apresentados nas duas seções anteriores podem ser generalizados. Para entender como esta generalização é feita, assuma que $\mathcal{H}$ denote o conjunto de vértices do hipercubo unitário, num espaço com dimensão n. Assuma, ainda, que X^i denote o conjunto de pontos em $\mathcal{H}$ que possuem exatamente i componentes iguais, onde $i \in \{1, \ldots, n\}$. Assim, temos que $|X^i| = 2^{n-i}$.

Note-se que o plano-de-corte esférico em (6.6) é uma desigualdade válida somente para os pontos de $\mathcal{H}$ que estão a uma distância de Hamming de no mínimo p, a partir de qualquer ponto do conjunto X^n.

Por outro lado, o plano-de-corte cilídrico em (6.10) é uma desigualdade válida somente para os pontos em $\mathcal{H}$ que estão a uma distância de no mínimo p, a partir de qualquer ponto pertencente ao conjunto X^{n-1}.

Portanto, em geral, os planos-de-corte geométricos correspondem a desigualdades válidas somente para pontos 0-1 que estão a uma distância de no mínimo p, a partir de qualquer ponto em X^k, onde X^k é um conjunto de pontos em $\mathcal{H}$ que difere em exatamente k componentes. Se $\overline{x}$ é um ponto em X^k, então esta desigualdade pode ser expressa por

$$\sum_{\substack{j=1 \\ j \notin \mathcal{K}}}^{n} (1 - 2\overline{x}_j) x_j \geq p - \sum_{\substack{j=1 \\ j \notin \mathcal{K}}} \overline{x}_j, \tag{6.12}$$

onde $\mathcal{K}$ é o conjunto de índices em $\{1, \ldots, n\}$, para os quais as componentes dos pontos em X^k são iguais.

6.5 Planos-de-corte canônicos

Nesta seção, estabelecemos uma relação entre os planos-de-corte geométricos e os planos-de-corte canônicos introduzidos por Balas e Jeroslow [5].

Mostramos que os planos-de-corte p-esférico e p-cilíndrico são casos especiais dos planos-de-corte canônicos. Mais do que isso, a partir da generalização dos planos-de-corte geométricos, podemos mostrar que os planos-de-corte canônicos e geométricos são realmente equivalentes. Portanto, o trabalho desenvolvido neste capítulo pode ser entendido como uma reinterpretação dos resultados originais de Balas e Jeroslow.

Considere o seguinte hipercubo unitário $\mathcal{H}_n$

$$\mathcal{H}_n = \{x \in \mathbb{R}^n \mid 0 \leq x_j \leq 1 \, , j \in N\}, \tag{6.13}$$

onde $N = \{1, \ldots, n\}$. Logo, cada vértice de $\mathcal{H}_n$ é um ponto 0-1 satisfazendo n desigualdades de (6.13) na igualdade.

Seja $\mathcal{H}$ o conjunto de vértices de $\mathcal{H}_n$. Uma face $\mathcal{F}^k$ de $\mathcal{H}_n$, com dimensão igual a k, onde $0 \leq k < n$, é um conjunto de pontos de $\mathcal{H}_n$ que satisfazem exatamente $n - k$ desigualdades de (6.13) na igualdade. Assim, todos os pontos em $\mathcal{H} \cap \mathcal{F}^k$ têm $n - k$ componentes idênticas àquelas associadas a $n - k$ equações $x_j = 0$ ou $x_j = 1$, que define a face $\mathcal{F}^k$.

Dado que $d \in \{0, 1, \ldots, n - k\}$, o hiperplano canônico de ordem k associado à face $\mathcal{F}^k$ de $\mathcal{H}_n$, denotado por $\mathcal{H}(\mathcal{F}^k)_d$, pode ser expresso por

$$\sum_{i \in N(\mathcal{F}^k)^+} x_i - \sum_{i \in N(\mathcal{F}^k)^-} x_i = |N(\mathcal{F}^k)^+| - d, \tag{6.14}$$

onde

$$N(\mathcal{F}^k)^+ = \{j \in N \mid x_j = 1 \text{ para todo } x \in V \cap \mathcal{F}^k\} \tag{6.15}$$

e

$$N(\mathcal{F}^k)^- = \{j \in N \mid x_j = 0 \text{ para todo } x \in V \cap \mathcal{F}^k\}. \tag{6.16}$$

Seja $\mathcal{H}(\mathcal{F}^k)_d^+$ um semi-espaço fechado, definido pelo hiperplano canônico $\mathcal{H}(\mathcal{F}^k)_d$, quando substituímos a igualdade na equação (6.14) por uma desigualdade da forma '$\leq$'. O semi-espaço $\mathcal{H}(\mathcal{F}^k)_d^+$ é denominado de *plano-de-corte canônico.*

Quando $k = 0$, consideramos uma face com dimensão igual a zero de $\mathcal{H}_n$, isto é, $\mathcal{F}^0$ coincide com um ponto $\overline{x}$ em $\mathcal{H}$. Neste caso, obtemos o plano-de-corte canônico $\mathcal{H}(\mathcal{F}^0)_d$,

$$\begin{aligned}
\sum_{i|\overline{x}_i=1} x_i - \sum_{i|\overline{x}_i=0} x_i &\leq \sum_{i|\overline{x}_i=1} \overline{x}_i - d \Longrightarrow \\
\sum_{i=1}^{n} (2\overline{x}_i - 1)x_i &\leq \sum_{i=1}^{n} \overline{x}_i - d \Longrightarrow \\
\sum_{i=1}^{n} (1 - 2\overline{x}_i)x_i &\geq d - \sum_{i=1}^{n} \overline{x}_i.
\end{aligned}$$

A desigualdade acima é precisamente o plano-de-corte d-esférico, descrito na seção 6.2.2, para o ponto $\overline{x}$.

Agora, suponha que $k = 1$ e considere a face de $\mathcal{H}_n$ com dimensão igual a 1, isto é, $\mathcal{F}^1$ coincide com o segmento de reta, de tamanho unitário, que une dois pontos adjacentes $\overline{x}^1$ and $\overline{x}^2$ de $\mathcal{H}$. Suponha que $\overline{x}^1$ e $\overline{x}^2$ diferem somente na componente k. Neste caso, obtemos o plano-de-corte canônico $\mathcal{H}(\mathcal{F}^1)_d$,

$$\begin{aligned}
\sum_{i|\overline{x}_i^1=\overline{x}_i^2=1} x_i - \sum_{i|\overline{x}_i^1=\overline{x}_i^2=0} x_i &\leq \sum_{i|\overline{x}_i^1=\overline{x}_i^2=1} \overline{x}_i^1 - d \Longrightarrow \\
\sum_{i|\overline{x}_i^1=\overline{x}_i^2} (2\overline{x}_i^1 - 1)x_i &\leq \sum_{i|\overline{x}_i^1=\overline{x}_i^2=1} \overline{x}_i^1 - d \Longrightarrow \\
\sum_{\substack{i=1 \\ i\neq k}}^{n} (1 - 2\overline{x}_i^1)x_i &\geq d - \sum_{\substack{i=1 \\ i\neq k}}^{n} \overline{x}_i^1.
\end{aligned}$$

Esta última desigualdade é precisamente um plano-de-corte d-cilíndrico para os pontos $\overline{x}^1$ e $\overline{x}^2$ descritos na seção 6.3.2.

Seguindo o raciocínio acima, podemos concluir que o plano-de-corte canônico para a face $\mathcal{F}^k$, definida para o plano-de-corte geométrico, com respeito ao hiperplano $\mathcal{H}(\mathcal{F}^k)_d^+$, é equivalente ao plano-de-corte geométrico, definido na desigualdade (6.12), para algum ponto $\overline{x} \in \mathcal{F}^k$, quando $d = p$.

6.6 Desigualdades de eliminação de subciclos e planos-de-corte geométricos

O propósito desta seção é mostrar que a desigualdade de eliminação de subciclos, definida para o problema do caixeiro viajante simétrico, corresponde a um plano-de-corte geométrico ou, de acordo com a seção anterior, a um plano-de-corte canônico.

Considere que $G = (V, E)$ seja um grafo não direcionado e que S seja um subconjunto de vértices de G, tal que $2 < |S| \leq |V|/2$. A desigualdade de *eliminação de subciclos* para o problema do caixeiro viajante simétrico é dada por

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad \emptyset \neq S \subset V, \tag{6.17}$$

onde $E(S)$ é o conjunto de arestas no grafo induzido por S em G (vide Jünger *et al.* [33]).

Agora, considere o conjunto X de todos os vetores em $\mathbb{R}^{|E|}$, cujas componentes associadas às arestas em $E(S)$ sejam iguais a 1. Em outras palavras, o

conjunto X corresponde ao conjunto de vértices do hipercubo, com dimensão igual a $|E|$, que fica na face definida pelas equações $x_e = 1$, para toda aresta $e \in E(S)$.

Essencialmente, a equação (6.17) nos mostra que todos os vetores de incidência que representam ciclos hamiltonianos em G, possuem uma distância de no mínimo

$$\binom{|E(S)|}{2} - (|S| - 1),$$

a partir de cada ponto em X. Esse valor é o caso geral (um grafo completo ou não). No caso de um grafo completo, seria a combinação de $|S|$ dois a dois.

Assuma que as arestas de $E(S)$ são aquelas que correspondem às primeiras $|E(S)|$ componentes dos vetores 0-1. Assim, se $\overline{x}$ é um ponto em X, a afirmação acima pode ser escrita em termos de um plano-de-corte geométrico,

$$\sum_{e \in E(S)} (1 - 2\overline{x}_e) x_e \geq \binom{|E(S)|}{2} - (|S| - 1) - \sum_{e \in E(S)} \overline{x}_e. \tag{6.18}$$

Entretanto, pela definição de X, temos que $\overline{x}_e = 1$ para toda aresta $e \in E(S)$. Portanto, a desigualdade (6.18) pode ser reescrita como

$$-\sum_{e \in E(S)} x_e \geq \binom{|E(S)|}{2} - (|S| - 1) - \binom{|E(S)|}{2},$$

ou, depois de simplificarmos os termos,

$$\sum_{e \in E(S)} x_e \leq |S| - 1,$$

que corresponde, precisamente, à desigualdade de eliminação de subciclos expressa pela equação (6.17).

Em geral, se P é um poliedro inteiro em $\mathbb{B}^n$ e $\sum_{j \in S} x_j \leq k$ é uma desigualdade válida para o poliedro P, onde $S \subseteq \{1, \ldots, n\}$, então esta desigualdade pode ser reescrita como um plano-de-corte geométrico,

$$\sum_{j \in S}(1 - 2\overline{x}_j)x_j \geq (n - k) - \sum_{j \in S}\overline{x}_j, \tag{6.19}$$

onde $\overline{x}$ corresponde ao vetor de incidência de S.

Isto inclui várias desigualdades conhecidas para problemas de otimização combinatória. Como exemplo destas desigualdades podemos mencionar as desigualdades de cobertura para o politopo da mochila (vide Balas e Jeroslow [5]).

Capítulo 7

Um algoritmo baseado em planos-de-corte geométricos para o PCVS

No capítulo anterior foi desenvolvida uma teoria para a geração de planos-de-corte genéricos para problemas lineares inteiros com variáveis binárias. Neste capítulo, utilizaremos estes planos-de-corte, para a resolução do problema do caixeiro viajante simétrico (PCVS), como parte da estrutura de um algoritmo *cut-and-branch*.

Como vimos no Capítulo 1, um dos problemas que surgem no projeto lógico de uma rede *backbone*, em particular, o problema de construção dos anéis *SONET* idênticos, corresponde à resolução de uma instância do PCVS.

Assim, um bom desempenho do algoritmo *cut-and-branch* na resolução do PCVS seria um indicativo de que podemos empregá-lo na resolução do problema de construção dos anéis *SONET* idênticos.

Inicialmente, apresentaremos a estrutura e os aspectos relevantes envolvidos com a implementação de um algoritmo *cut-and-branch*. Apresentaremos, ainda, como é possível o emprego dos planos-de-corte geométricos, mais especificamente, dos planos-de-corte p-esféricos, na representação de desigualdades de eliminação de subciclos. Além disso, traçaremos uma relação entre os movimentos de melhoria propostos por Lin e Kernighan e os planos-de-corte p-esféricos.

Para finalizar, apresentaremos o algoritmo *cut-and-branch* implementado para o problema do caixeiro viajante simétrico e relataremos alguns resulta-

dos computacionais obtidos com o emprego deste algoritmo.

7.1 Estrutura e aspectos de um algoritmo *cut-and-branch*

Um algoritmo *cut-and-branch* é constituído por duas fases: um algoritmo de planos-de-corte e um procedimento *branch-and-bound*. Neste tipo de algoritmo, os planos-de-corte são gerados apenas no nó raiz da árvore *branch-and-bound* (vide Johnson *et al.* [32]).

Os primeiros trabalhos que fizeram uso de um algoritmo *cut-and-branch* foram os de Crowder *et al.* [13], para problemas de programação inteira 0-1, e Van Roy e Wolsey [62], para problemas de programação inteira mista. Recentemente, Lim e Kim utilizaram um algoritmo *cut-and-branch* para a resolução do problema de localização de fábricas (vide [41, 42]).

A estrutura de um algoritmo *cut-and-branch*, para problemas de minimização, é apresentada no Algoritmo 6. Antes, assuma a seguinte notação. Sejam RL o problema de programação linear relaxado, $\mathcal{N}$ o nó raiz da árvore de *branch-and-bound* contendo o problema RL, *num_cortes* o número de planos-de-corte gerados, e $\overline{x}$ a solução atual, obtida a partir do problema RL. Considere, ainda, que $\overline{z}$ seja o valor da melhor solução obtida para um dado problema, e $\underline{z}$ o limite inferior obtido com a resolução do problema RL.

Nas linhas 2-3, fazemos algumas inicializações. Na linha 6, resolvemos o problema RL, através de algum pacote de otimização. Na linha 9, procuramos por planos-de-corte violados pela solução atual $\overline{x}$. Nas linhas 10-12, verificamos se foi ou não possível encontrar algum plano-de-corte. Na linha 16, empregamos o procedimento *branch-and-bound* para provar a otimalidade do problema RL atual. Já nas linhas 19-25, testamos as condições de término do algoritmo e atualizamos o valor de $\overline{z}$. Observe que o algoritmo *cut-and-branch* terminará em duas situações: ou quando atingirmos um número máximo de planos-de-corte, denotado por *num_max_cortes*, ou quando provarmos a otimalidade do problema que estamos resolvendo.

Algoritmo 6 Algoritmo informal para um algoritmo *cut-and-branch*

1: /* *Inicialização* */
2: Faça $\overline{z} = +\infty$ e $\underline{z} = -\infty$
3: Defina o problema RL inicial
4:
5: /* *Relaxação* */
6: Resolva o problema RL e obtenha a solução $\overline{x}$
7:
8: /* *Separação* */
9: Procure por planos-de-corte violados por $\overline{x}$
10: **Se** forem encontrados planos-de-corte **Então**
11: Adicione-os ao problema RL atual
12: Incremente o valor de *num_cortes*
13: **Fim-Se**
14:
15: /* *Branch-and-bound* */
16: Resolva o problema RL atual do nó raiz $\mathcal{N}$ e obtenha $\underline{z}$
17:
18: /* *Atualização/Término* */
19: **Se** $(\overline{z} - \underline{z} < \epsilon)$ ou $(num_cortes = num_max_cortes)$ **Então**
20: PARE
21: **Fim-Se**
22: **Se** $(\overline{z} > \underline{z})$ **Então**
23: $\overline{z} := \underline{z}$
24: Retorne ao passo (3)
25: **Fim-Se**

7.2 Emprego dos planos-de-corte p-esféricos

Nesta seção, mostraremos que os planos-de-corte p-esféricos podem ser utilizados no algoritmo *cut-and-branch*, quando estamos interessados na resolução do PCVS. Antes, assuma que $S \subset V_n$ seja um conjunto de vértices quaisquer, $E(S)$ denote o conjunto de arestas com ambos os extremos em S, e que $E(V \setminus S)$ denote o conjunto de arestas com um extremo em V e o outro extremo em S.

Considere a relaxação *2-matching* apresentada na seção 7.4.1. Podemos observar que para $n = 3, 4$ e 5, a solução obtida corresponderá a um único ciclo.

Assuma agora que $n \geq 6$. Na Figura 7.1(a), temos um exemplo de um grafo completo com $n = 6$ e $m = 15$. Uma possível solução obtida para este grafo é apresentada na Figura 7.1(b). Nota-se a existência de dois ciclos: $C_1 = \{(1,2),(1,3),(2,3)\}$ e $C_2 = \{(4,5),(4,6),(5,6)\}$. Logo, o ponto x associado aos valores 0-1 das arestas no grafo da Figura 7.1 corresponde a $x_e = 1$, para toda aresta $e \in E(C_1)$ ou $e \in E(C_2)$, e $x_e = 0$, em caso contrário.

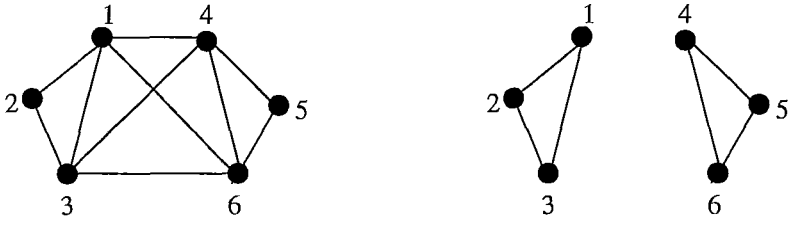

Figura 7.1: (a) Grafo $K_n = (V_n, E_n)$. (b) Solução obtida com relaxação 2-*matching* do PCVS para o grafo K_n.

Aplicando a desigualdade (7.7), desigualdade de eliminação de subciclos, no ponto x definido acima, obtemos as seguintes desigualdades violadas:

$$x_{12} + x_{13} + x_{23} \leq 2 \tag{7.1}$$

e

$$x_{45} + x_{46} + x_{56} \leq 2. \tag{7.2}$$

Multiplicando as desigualdades (7.1) e (7.2) por 4, e as desigualdades de grau, associadas a cada vértice de V, por -1, é possível obtermos, através de uma combinação linear, a seguinte desigualdade:

$$2 \times \left(\sum_{e \in E(C_1)} x_e + \sum_{e \in E(C_2)} x_e - \sum_{e \in E(V \setminus (C_1)} x_e - \sum_{e \in E(V \setminus (C_2)} x_e \right) \leq 4.$$

Se dividirmos a desigualdade acima por 2, obteremos um plano-de-corte 4-esférico,

$$x_{12} + x_{13} - x_{14} - x_{15} - x_{16} + x_{23} - x_{24} - x_{25} - x_{26} - \\ -x_{34} - x_{35} - x_{36} + x_{45} + x_{46} + x_{56} \leq 2. \qquad (7.3)$$

Observe que as arestas com coeficientes iguais a $+1$ na desigualdade (7.3) correspondem às arestas que definem as desigualdades (7.1) e (7.2), isto é, as arestas com valores iguais a 1, na solução x. Por outro lado, as arestas com coeficientes iguais a -1 na desigualdade (7.3) correspondem às arestas que tiveram seus coeficientes modificados para -1 nas desigualdades de grau, isto é, as arestas com valores iguais a zero, na solução x. Assim, um plano-de-corte p-esférico pode ser obtido através de uma combinação linear das desigualdades de eliminação de subciclos e de grau.

Essa observação reforça a afirmação que fizemos na seção 6.6 sobre os planos-de-corte geométricos e as desigualdades de eliminação de subciclos, ou seja, que dada uma solução $x \in \mathbb{B}^n$, sempre podemos representar as desigualdades de eliminação de subciclos através dos planos-de-corte p-esféricos.

Na próxima seção, faremos uma relação entre os movimentos de melhoria propostos por Lin e Kernighan [43] e os planos-de-corte p-esféricos.

7.3 Heurística de Lin-Kernighan e planos-de-corte p-esféricos

Os movimentos de melhoria propostos por Lin-Kernighan são comumente utilizados em heurísticas empregadas na resolução do problema do caixeiro

viajante simétrico. Estes movimentos podem ser realizados ou sob os vértices, ou sob as arestas, e se baseiam na troca de arestas (ou vértices) que pertencem à solução atual.

Uma analogia a estes movimentos de melhoria pode ser feita com o emprego dos planos-de-corte p-esféricos definidos no Capítulo 6. Seja $\overline{x}$ uma solução obtida para o problema do caixeiro viajante simétrico, a partir de uma formulação que não possui as desigualdades de eliminação de subciclos. Considere, ainda, que C representa o conjunto de ciclos disjuntos presentes na solução $\overline{x}$.

Assumindo que $|C| = c$, podemos obter o seguinte plano-de-corte p-esférico,

$$\sum_{j=1}^{|E|}(x_j - \overline{x}_j)^2 \geq p = 2 \times c, \tag{7.4}$$

onde p é a distância mínima de $\overline{x}$, para que se possa encontrar uma solução viável (um único ciclo) para o problema do caixeiro viajante simétrico.

Vejamos, agora, um exemplo da aplicação da desigualdade (6.6). Considere o grafo completo da Figura 7.1(a), onde $n = 6$ e $m = 15$. Como visto anteriormente, uma possível solução $\overline{x}$ é apresentada na Figura 7.1(b). Aplicando a desigualdade (7.4) à solução (ou ponto) $\overline{x}$, com $c = 2$ e $p = 2 \times 2 = 4$, obtemos o seguinte plano-de-corte 4-esférico,

$$\begin{aligned} &\sum_{j=1}^{15}(1 - 2\overline{x}_j)x_j \geq 4 - \sum_{j=1}^{15}\overline{x}_j \Rightarrow \\ \Rightarrow\ & x_{12} + x_{13} - x_{14} - x_{15} - x_{16} + x_{23} - x_{24} - x_{25} - \\ & -x_{26} - x_{34} - x_{35} - x_{36} + x_{45} + x_{46} + x_{56} \leq 2, \end{aligned} \tag{7.5}$$

que possibilita a eliminação de todos os pontos pertecentes ao conjunto $W_4(\overline{x})$, isto é, a eliminação de pontos que representam soluções inviáveis para o PCVS.

A Figura 7.2 mostra um exemplo do emprego da desigualdade (6.6), obtida a partir da solução $\overline{x}$. Como existe uma relação entre os planos-de-corte

p-esféricos e as desigualdades de eliminação de subciclos, conforme visto na seção 6.6, pode-se observar que o acréscimo da desigualdade (7.5) à relaxação 2-*matching* poderá resultar em uma solução viável para o problema do caixeiro viajante simétrico.

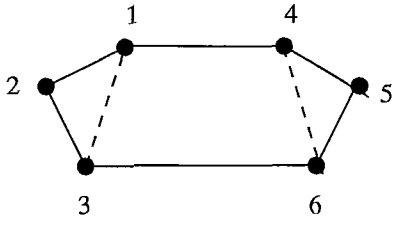

Figura 7.2: Um exemplo de aplicação do plano-de-corte 4-esférico $x_{12}+x_{13}-x_{14}-x_{15}-x_{16}+x_{23}-x_{24}-x_{25}-x_{26}-x_{34}-x_{35}-x_{36}+x_{45}+x_{46}+x_{56} \leq 2$ na solução $\overline{x}$.

7.4 Visão geral do algoritmo *cut-and-branch*

A implementação de um algoritmo *cut-and-branch* envolve muitas questões. Algumas dessas questões foram apresentadas na seção 7.1. Nesta seção, apresentamos o algoritmo *cut-and-branch* que propusemos para a resolução de pequenas instâncias do PCVS.

7.4.1 Modelo matemático inicial

Consideramos que o leitor tenha algum conhecimento básico sobre o problema do caixeiro viajante simétrico. Seja $K_n = (V_n, E_n)$ um grafo completo não dirigido, com $|V_n| = n$ e $|E_n| = m = n(n-1)/2$. No PCVS, deseja-se encontrar um ciclo hamiltoniano de comprimento mínimo, isto é, um ciclo que contenha cada vértice uma única vez e que possua comprimento mínimo.

O PCVS pode ser formulado como um problema de programação linear inteira (PPLI). Para tanto, considere a seguinte variável de decisão: $x_e = 1$, se somente se a aresta e pertence ao ciclo hamiltoniano, e $x_e = 0$, caso contrário.

Além disso, assuma que c_e seja um valor inteiro associado a toda aresta $e \in E_n$. Considere, ainda, que $\delta(\{i\})$ denote o conjunto de arestas incidentes

ao vértice i. Apresentada a definição das variáveis e a notação a ser adotada, o PCVS pode expresso através do seguinte PPLI 0-1:

(P7)

$$\text{minimize} \sum_{e \in E} c_e x_e,$$

sujeito a:

$$\sum_{e \in \delta(\{i\})} x_e = 2, \qquad i = 1, \ldots, n, \tag{7.6}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1, \qquad \text{para todo } \emptyset \subset S \subset V_n, \tag{7.7}$$

$$x_e \in \{0, 1\}, \qquad \forall e \in E_n. \tag{7.8}$$

As restrições (7.6) são denominadas desigualdades 2-*matching* e garantem que cada vértice seja visitado uma única vez. As restrições (7.7) são denominadas de desigualdades de eliminação de subciclos e impedem a formação de ciclos disjuntos. Por último, as restrições (7.8) são as restrições de integralidade.

A formulação obtida apenas pelas restrições (7.6) e (7.8) é denominada relaxação 2-*matching* do problema do caixeiro viajante simétrico; as soluções resultantes dela são denominadas 2-*matchings*. Tais soluções correspondem a ciclos disjuntos.

O problema *RL* inicial corresponderá à relaxação 2-*matching*, como descrita acima. Com a resolução deste problema, obteremos um limite inferior inicial.

7.4.2 Rotinas de separação

Nesta seção, descreveremos a rotina de separação utilizada na fase de planos-de-corte, para identificar as desigualdades violadas. Os planos-de-corte p-esféricos são separados através de um algoritmo semelhante ao algoritmo para encontrar as componentes conexas de um grafo (vide Cormen *et al.* [11]). A complexidade dessa rotina é $O(n^2)$.

A entrada para a rotina de separação corresponde à solução atual $\overline{x}$ do problema *RL*, enquanto que a saída corresponderá a um plano-de-corte p-

esférico violado por esta solução. Tal desigualdade será adicionada ao problema RL atual do nó raiz $\mathcal{N}$ e, assim, obteremos um novo problema de programação linear inteira.

No Algoritmo 7, apresentamos um algoritmo informal para a rotina de separação. Seja gr um vetor de n posições e mt uma matriz de ordem $m \times 2$. A matriz mt indicará, para cada vértice $v \in V_n$, os dois vértices adjacentes a ele na solução $\overline{x}$.

Algoritmo 7 Algoritmo informal para a rotina de separação

1: **Para todo** $v \in V$ **faça**
2: $\quad gr[v] = 0$
3: **Fim-Para**
4: **Para todo** $(u, v) \in E$ **faça**
5: $\quad gr[u]$ ++
6: $\quad gr[v]$ ++
7: $\quad mt[u][gr[u]] = v$
8: $\quad mt[v][gr[v]] = u$
9: **Fim-Para**
10: Obtenha o número de ciclos a partir da matriz mt
11: Construa o plano-de-corte p-esférico

Nas linhas 1-3, inicializamos o grau de cada vértice. Nas linhas 4-9, identificamos os dois vértices adjacentes a cada vértice da solução $\overline{x}$. Por último, nas linhas 10-11, obtemos os ciclos presentes na solução $\overline{x}$. Para isso, percorremos a matriz mt e, em seguida, construímos o plano-de-corte p-esférico definido por $\overline{x}$.

7.4.3 Procedimento *branch-and-bound*

Após o acréscimo dos planos-de-corte p-esféricos ao problema RL atual do nó raiz $\mathcal{N}$, executamos um procedimento *branch-and-bound* até atingirmos a otimalidade do problema RL.

Para a execução do procedimento *branch-and-bound*, utilizou-se o resolvedor linear (inteiro) XPRESS. O valor empregado para o número máximo de nós que podem ser abertos durante a execução do procedimento *branch-and-bound* foi igual a 10^7, ou seja, o valor padrão deste resolvedor.

7.5 Resultados computacionais

O algoritmo *cut-and-branch* descrito na seção anterior foi implementado na linguagem de programação `C`. Os experimentos computacionais foram executados em um computador `AMD K6 450MHz` com `256MBytes` de memória `RAM`.

Para investigarmos o desempenho do algoritmo *cut-and-branch* proposto, bem como a eficiência dos planos-de-corte p-esféricos, na resolução do PCVS, desabilitamos a geração automática de planos-de-corte do XPRESS 12.05 (vide [14]).

7.5.1 Descrição das instâncias

Foram compiladas algumas instâncias da biblioteca *TSPLIB* [61]. Este conjunto de instâncias é mostrado na Tabela 7.1. A primeira coluna se refere a instância testada, a segunda, o número de vértices, e a terceira, e última coluna, indica o número de arestas.

Nome	n	m
gr17	17	136
gr24	24	276
gr48	48	1128
bays29	29	406
bayg29	29	406
hk48	48	1128
swiss42	42	861

Tabela 7.1: Instâncias utilizadas nos experimentos computacionais.

7.5.2 Experimentos

Nesta seção, relatamos os experimentos computacionais obtidos com o emprego dos planos-de-corte p-esféricos em uma estrutura *cut-and-branch*.

Na Tabela 7.2, apresentamos um comparativo entre a qualidade da solução obtida na primeira relaxação linear, isto é, a relaxação 2-*matching* inicial, com o valor da solução ótima de cada instância testada.

A primeira coluna da tabela indica a instância testada; a segunda coluna indica o valor ótimo, denotado por z^*, da instância; a terceira coluna, o valor obtido na relaxação linear, denotado por z_{rl}; e, por último, a quarta coluna indica a diferença percentual entre o valor da relaxação linear e o valor ótimo, ou seja,

$$\Delta(z^*, z_{rl}) = \frac{(z^* - z_{rl})}{z^*} \times 100.$$

Nome	z^*	z_{rl}	$\Delta(z^*, z_{rl})$
gr17	2085	1684	19.23
gr24	1272	1227	3.54
gr48	5046	4807	4.74
bays29	2020	1958.5	3.04
bayg29	1610	1548	3.85
hk48	11461	11197	2.31
swiss42	1273	1228.5	3.50

Tabela 7.2: Qualidade da solução obtida na relaxação inicial.

O critério de parada adotado para o algoritmo *cut-and-branch* está relacionado ao número máximo de planos-de-corte, denotado por *num_max_cortes*, que podemos gerar durante a execução do algoritmo. Foram adotados os seguintes valores para este parâmetro duante os experimentos: 100, 200 e 300.

Nas Tabelas 7.3, 7.4 e 7.5, apresentamos os resultados obtidos para *num_max_cortes* igual a 100, 200 e 300, respectivamente. As colunas destas três tabelas são descritas em seguida. A primeira coluna indica a instância testada; a segunda, terceira e quarta colunas indicam, respectivamente, o número mínimo, máximo e total de nós abertos na árvore *branch-and-bound*; a quinta coluna indica o limite inferior, denotado por z_{lb}, obtido pelo algoritmo *cut-and-branch*; a sexta coluna, a diferença percentual entre a solução ótima z^* e a solução obtida z_{lb}, isto é,

$$\Delta(z^*, z_{lb}) = \frac{(z^* - z_{lb})}{z^*} \times 100;$$

por fim, a sétima coluna indica o tempo computacional empregado na resolução da instância pelo algoritmo *cut-and-branch*. Esse tempo está expresso em segundos.

Como dito anteriormente não fizemos uso dos planos-de-corte embutidos no resolvedor linear XPRESS. Durante a rotina de separação do algoritmo *cut-and-branch*, permitimos, a cada iteração deste algoritmo, apenas a geração de um plano-de-corte p-esférico violado.

Observando as Tabelas 7.3, 7.4 e 7.5, nota-se que à medida que aumentamos o número máximo de planos-de-corte a serem gerados, obtemos uma melhoria na qualidade das soluções das instâncias testadas. Por exemplo, para a instância `gr24`, foi possível provarmos a sua otimalidade com um tempo inferior a 6 minutos e um número de planos-de-corte igual a 200. Devido a isso, não testamos essa instância para *num_max_cortes* igual a 300.

Em outras quatro instâncias testadas, `bays29`, `bayg29`, `hk48` e `swiss42`, os valores encontrados para os limites inferiores estão a menos de 1% do valor ótimo. Apenas para as instâncias `gr17` e `gr48`, a diferença percentual foi maior do que 2% do valor ótimo.

Entretanto, esta qualidade obtida nas soluções se reflete no número de nós da árvore *branch-and-bound* e no tempo computacional. Nota-se que o número de nós é praticamente cinco vezes maior, quando aumentamos o número de planos-de-corte. O mesmo se pode observar com relação ao tempo computacional, que chega a apresentar um valor quase dez vezes maior.

	# Nds					
Nome	min	max	total	z_{lb}	$\Delta(z^*, z_{lb})$	Tempo
gr17	1	1810	96232	1960	5.99	288
gr24	5	716	34977	1266	0.47	113
gr48	19	2757	149267	4886	3.17	769
bays29	3	884	46396	1995	1.24	289
bayg29	7	950	43688	1585	1.55	277
hk48	1	4016	187667	11365	0.84	1051
swiss42	47	3242	147044	1254	1.49	1580

Tabela 7.3: Resultados computacionais obtidos para # Cortes = 100.

	# Nds					
Nome	min	max	total	z_{lb}	$\Delta(z^*, z_{lb})$	Tempo
gr17	1	3548	365894	1986	4.75	1966
gr24	5	1245	131242	1272	0.00	717
gr48	19	5413	545236	4898	2.93	4654
bays29	3	1798	173347	2004	0.79	1796
bayg29	7	1900	186974	1592	1.12	1941
hk48	1	8566	817971	11392	0.60	7868
swiss42	47	7369	689379	1259	1.09	14193

Tabela 7.4: Resultados computacionais obtidos para # Cortes = 200.

	# Nds					
Nome	min	max	total	z_{lb}	$\Delta(z^*, z_{lb})$	Tempo
gr17	1	5187	788931	2001	4.03	6631
gr48	19	9058	1234107	4906	2.77	26412
bays29	3	2867	405250	2008	0.59	5998
bayg29	7	3042	431227	1596	0.87	6277
hk48	1	13258	1907021	11408	0.46	18396
swiss42	47	13025	1734953	1262	0.86	48957

Tabela 7.5: Resultados computacionais obtidos para # Cortes = 300.

7.6 Considerações

Os resultados computacionais obtidos com os experimentos demonstraram que os planos-de-corte p-esféricos, em uma estrutura *cut-and-branch*, fornecem limites inferiores de boa qualidade para o problema do caixeiro viajante simétrico. Na maioria das instâncias testadas, a diferença percentual obtida foi de no máximo 4.03%. Além disso, os resultados apresentaram um tempo computacional bastante razoável.

Com base nestes dois aspectos, propomos o emprego dos planos-de-corte p-esféricos na resolução do problema de construção dos anéis *SONET* idênticos que surge no projeto lógico de uma rede *backbone*.

Uma outra extensão do uso dos planos-de-corte p-esféricos seria provar a otimalidade de soluções do problema do caixeiro viajante simétrico. Seja $\overline{x}$ a solução encontrada por uma heurística aplicada ao PCVS e $\overline{z}$ o valor da função associado à solução $\overline{x}$. Para sabermos se $\overline{x}$ é uma solução ótima, poderemos introduzir um plano-de-corte p-esférico e,liminando $\overline{x}$. Se a nova

solução encontrada for maior ou igual a $\overline{z}$, então $\overline{x}$ será uma solução ótima.

Além disso, podemos também propor a utilização deste algoritmo em uma fase de pré-processamento de algoritmos que empregam planos-de-corte na resolução do problema do caixeiro viajante simétrico. Com isto, poderíamos tentar melhorar o desempenho destes algoritmos, e assim procurar obter bons limites inferiores para o PCVS.

Capítulo 8

Conclusões e trabalhos futuros

8.1 Contribuições

Nesta tese, estudamos problemas que ocorrem no planejamento de redes de telecomunicações. Em particular, demonstramos interesse no planejamento de uma rede *bakcbone*. O planejamento desta rede consiste em determinar como devem ser efetuadas as conexões entre as suas localidades de modo a minimizar os custos de instalação e ainda atender a um conjunto de restrições.

A abordagem de resolução dos problemas que surgem durante o planejamento de uma rede *backbone* é, usualmente, dividida em duas etapas: projeto físico e lógico. O problema presente no projeto físico é denominado de problema de atribuição de localidades a anéis, enquanto que um dos problemas que surge no projeto lógico, relacionado com o problema do caixeiro viajante simétrico, é denominado de problema de construção dos anéis *SONET* idênticos.

Para o problema de atribuição de localidades a anéis (SRAP), propomos duas novas formulações inteiras 0-1, denotadas por (P2) e (P3), e discutidas nos Capítulos 2 e 5, respectivamente. Ambas as formulações obtiveram resultados computacionais muito melhores do que aqueles obtidos pela formulação existente na literatura. Estes resultados, em algumas instâncias testadas, representam uma melhoria de horas para segundos. Este desempemho computacional atesta que as formulações que propusemos podem ser empregadas de maneira mais eficiente na resolução do problema de atribuição

de localidades a anéis.

Estudamos uma abordagem exata do problema de atribuição de localidades a anéis baseada em planos-de-corte. Assim, realizamos no Capítulo 3 uma investigação inicial da estrutura facial do politopo $\mathcal{Q}$ associado ao SRAP.

Para a formulação de empacotamento (P2), provamos a dimensão do politopo $\mathcal{Q}$ e procuramos caracterizar a força das desigualdades presentes nesta formulação, ou seja, provar quais dentre estas desigualdades são definidoras de facetas para $\mathcal{Q}$. Além destes resultados básicos, definimos novas classes de desigualdades válidas para o politopo $\mathcal{Q}$ dadas pelas equações (3.3) e (3.4).

Também apresentamos condições para que algumas destas desigualdades sejam definidoras de facetas para $\mathcal{Q}$. Fomos, ainda, capazes de encontrar novas classes de desigualdades definidoras de facetas para o politopo $\mathcal{Q}$ que foram obtidas a partir de soluções fracionárias ótimas $\mathcal{X}^*$ (veja as equações (3.1), (3.2) e (3.5)).

Os resultados obtidos com o emprego das classes de desigualdades definidas para o politopo $\mathcal{Q}$, indicaram algumas dificuldades na resolução de instâncias de médio porte ($n = 25$). Isto foi constatado até mesmo com a melhor das estratégias propostas: a estratégia F1 do Capítulo 3.

Na maioria das instâncias, verificou-se a exploração de um elevado número de nós na árvore de enumeração, uma diferença percentual muito grande entre o valor da relaxação linear e o ótimo inteiro, e a interrupção da execução da estratégia F1 pela limitação de tempo. Constatamos, a partir destas dificuldades, que a questão da simetria apresentada por esta formulação do problema de atribuição de localidades a anéis impactava sobremaneira o desempenho do algoritmo *branch-and-bound.*

Diante das dificuldades expostas acima, dedicamo-nos ao estudo de como poderíamos quebrar a simetria entre as soluções do SRAP, ou seja, reformular este problema através da inclusão de desigualdades lineares à formulação de empacotamento (P2). As desigualdades que definimos, visando superar aquelas dificuldades, foram apresentadas no Capítulo 3 (veja as equações (3.6) e (3.7)).

Estas desigualdades deram lugar a duas estratégias de eliminação de si-

metria denotadas respectivamente por S1 e S2. Estas tentativas de redução da simetria do modelo melhoraram sensivelmente o desempenho computacional se compararmos estes resultados com aqueles obtidos apenas com a formulação de empacotamento (P2) ou com a estratégia F1. Em algumas instâncias o tempo de computação chega a ser nove vezes menor, como por exemplo, no caso da instância `gl_15_1`, ou até vinte vezes menor , como é o caso da instância `rl_25_8`.

Em alguns casos a reformulação do modelo de empacotamento foi capaz de reduzir, através da estratégia S1, a diferença entre o valor da relaxação linear e o ótimo inteiro, e a quantidade de nós enumerados na árvore *branch-and-bound*. Entretanto, o tempo computacional ainda continuava elevado, comprometendo a resolução, de forma exata, de instâncias de médio ($n = 25$) porte do problema de atribuição de localidade a anéis. Este fraco desempenho ainda estava relacionado à questão da simetria das soluções, que não tinha sido completamente solucionada. Procuramos, então, superar este obstáculo, através de uma reformulação do SRAP e do uso da técnica de geração de colunas.

Para fazermos uso da técnica de geração de colunas, descrevemos no Capítulo 5 uma nova reformulação do problema de atribuição de localidades a anéis. Nesta reformulação, denotada por (P3), o SRAP foi definido como sendo um problema de *set packing*. Os resultados obtidos com esta nova formulação foram bastante expressivos. Conseguimos resolver instâncias de pequeno e médio porte em menos de 10 minutos, e em algumas instâncias, como por exemplo, `gl_15_4`, `gh_25_2` e `rh_25_9`, estes tempos computacionias foram inferiores a 1 minuto. Se compararmos estes tempos com aqueles obtidos apenas com o emprego da formulação de empacotamento (P2), o ganho foi extraordinário.

Com base nesta nova formulação, definimos um problema mestre, como sendo um problema de *set partitioning*, e um problema auxiliar, como sendo um problema de programação quadrática máxima. Após alguns experimentos computacionais com o problema auxiliar, concluímos que seria interessante estudar a envoltória convexa das soluções inteiras do problema auxiliar, ou

seja, caracterizar a estrutura facial do politopo $\mathcal{P}'$ associado a este problema.

Além de provarmos a dimensão do politopo $\mathcal{P}'$, provamos quais desigualdades pertencentes ao modelo, que define o problema auxiliar, são definidoras de facetas para $\mathcal{P}'$. Ainda, neste estudo inicial, conseguimos obter algumas desigualdades válidas e estabelecer condições para que elas definam facetas para o politopo $\mathcal{P}'$(vide os Teoremas 5.4, 5.7, 5.8 e 5.9 da seção 5.4).

Após o estudo poliédrico preliminar do problema auxiliar, resolvemos implementar um algoritmo *branch-and-price*, visto que o foco desta parte do trabalho estava na resolução de forma exata do problema de atribuição de localidades a anéis.

Os resultados obtidos com o emprego algoritmo *branch-and-price* mostraram a viabilidade de se resolver, de forma exata, instâncias de médio e grande porte do problema de atribuição de localidades a anéis, através desta técnica. Os experimentos computacionais evidenciaram que o algoritmo *branch-and-price* é bastante competitivo, visto que para a maioria das instâncias testadas o tempo computacional gasto ficou abaixo de 4 minutos, sempre obtendo uma melhoria significativa neste tempo, se compararmos com o tempo gasto pela formulação (P1). Por exemplo, para as instâncias `gl_15_1`, `gl_15_4`, `gl_15_7` e `gl_15_9` a melhoria no tempo computacional foi de 1362, 1072, 4177 e 2976 vezes, respectivamente. Além disso, em 31 instâncias das 341 instâncias testadas, foi possível provar a otimalidade daquelas instâncias ainda no primeiro nodo da árvore *branch-and-bound.*

Feita esta análise sobre as contribuições que fizemos para o problema de atribuição de localidades a anéis nesta tese, iniciamos a análise dos nossos resultados para o outro problema que surge no projeto lógico e que foi tratado neste trabalho.

No estudo do problema de construção dos anéis *SONET* idênticos, vimos que o projeto lógico de uma rede *backbone* corresponde à resolução de uma instância do problema do caixeiro viajante simétrico. Inicialmente, propusemos, no Capítulo 6, uma família de planos-de-corte para programação linear inteira 0-1, que pode ser aplicada na resolução do problema do caixeiro viajante simétrico.

Tais planos-de-corte apresentam um caráter geométrico e, por isso, foram denominados de planos-de-corte geométricos. Os dois tipos de planos-de-corte estudados foram denominados de planos-de-corte esféricos e cilíndricos. Estes planos-de-corte foram discutidos em detalhes nas seções 6.2 e 6.3 do Capítulo 6.

Estas desigualdades ao invés de eliminarem soluções fracionárias, eles procuravam eliminar soluções inteiras inviáveis de problemas de programação linear inteira 0-1. Mostramos, ainda, neste trabalho que existe uma correspondência de um para um entre os planos-de-corte geométricos e os planos-de-corte canônicos propostos por Balas e Jeroslow em [5].

Por último, propusemos um algoritmo *cut-and-branch* para a resolução de pequenas instâncias do problema do caixeiro viajante simétrico. Este algoritmo fez uso dos planos-de-corte p-esféricos. Os resultados computacionais indicaram a viabilidade do emprego deste algoritmo na obtenção de limites inferiores de boa qualidade. Na maioria das instâncias testadas, a diferença percentual obtida foi, no máximo, igual a 4.03%.

Com base no sucesso desta iniciativa, propomos o emprego dos planos-de-corte p-esféricos na resolução do problema de construção de anéis *SONET* idênticos. Uma outra aplicabilidade do uso dos planos-de-corte p-esféricos seria tentar provar a otimalidade de soluções do problema do caixeiro viajante simétrico.

8.2 Direções futuras

Gostaríamos de deixar algumas direções futuras de pesquisa para a resolução dos problemas que surgem no planejamento da rede *backbone* que tratamos neste trabalho.

Para o problema de atribuição de localidades a anéis, podemos dividir estas direções sob dois pontos de vista: o prático e o teórico, ou seja, melhorias na implementação do algoritmo *branch-and-price* e um estudo do politopo $\mathcal{P}'$associado ao problema auxiliar.

Do ponto de vista computacional, poder-se-ia pensar na implementação de outras heurísticas para, assim, tentarmos melhorar o valor do limitante su-

perior. Isto poderia nos trazer uma diminuição no número de nós abertos na árvore *branch-and-bound*, e conseqüentemente uma melhoria no desempenho computacional.

Um outro aspecto que pode ser explorado na implementação do algoritmo *branch-and-price* é o uso da técnica de relaxação Lagrangeana para obter melhores limitantes inferiores a cada iteração deste algoritmo (veja Farley [18], Lasdon [38] e, Vanderbeck e Wolsey [74]).

Já sob o aspecto teórico, uma direção futura para o estudo do problema de atribuição de localidades a anéis seria continuar o estudo de simetria das soluções deste problema. Além disso, podemos pensar também no emprego da formulação (P3) acrescida de futuras desigualdades de simetria. Isto poderia acarretar numa melhoria no desempenho computacional desta formulação.

Um outro ponto que pode ser explorado, com relação ao aspecto teórico é continuarmos com a caracterização poliédrica do problema auxiliar. Antes, porém, devemos realizar um estudo mais detalhado entre a relação do politopo $\mathcal{P}'$ com os politopos $\mathcal{P}^{\mathcal{C}}$ e $\mathcal{P}^{\mathcal{B}}$. Um estudo inicial desta relação já foi apresentado neste trabalho. Este estudo poderá nos proporcionar um ótimo desempenho computacional na resolução do problema auxiliar, à medida que aumentamos o tamanho das instâncias.

Além disso, podemos ainda propor a definição de outras formulações inteiras para o SRAP. Por exemplo, como os agrupamentos possuem uma capacidade máxima em comum, ou seja, o valor B, pode-se pensar em duas outras formulações. Na primeira, substituiríamos a restrição (2.14) da formulação (P2) por:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv}p_{uvi} + \sum_{u=1}^{n}\sum_{v=1}^{n} d_{uv}z_{uvi} - By_i \leq 0, \quad \forall i \in V.$$

Já na segunda formulação, substituiríamos a restrição (5.7) da formulação (P3) por:

$$\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv} f_{uvi} - By_i \leq 0, \quad \forall i \in V.$$

Este tipo de restrição, onde as capacidades são estabelecidas para conjuntos idênticos, ocorre em problemas de localização de facilidades.

Uma terceira formulação linear inteira 0-1, derivada do trabalho de Aringhieri e Dell'Amico [3], é apresentada em seguida:

$$\text{maximize } \sum_{u=1}^{n}\sum_{i=1}^{n} \Theta_u x_{ui} - \sum_{i=1}^{n} y_i,$$

sujeito a:

$$\sum_{u=1}^{n} K_u x_{ui} - \sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv} p_{uvi} - By_i \leq 0, \qquad \forall i \in V, \tag{8.1}$$

$$\frac{1}{2}\sum_{u=1}^{n} K_u x_{ui} - \sum_{i=1}^{n}\sum_{u=1}^{n-1}\sum_{v=u+1}^{n} d_{uv} p_{uvi} \leq B, \tag{8.2}$$

$$\sum_{i=1}^{n} x_{ui} \leq 1, \qquad \forall u \in V,$$

$$x_{ui} - y_i \leq 0, \qquad \forall u \in V, i \in V,$$

$$p_{uvi} - x_{ui} \leq 0, \qquad \forall u, v \in V, u < v, \ \forall i \in V,$$

$$p_{uvi} - x_{vi} \leq 0, \qquad \forall u, v \in V, u < v, \ \forall i \in V,$$

$$x_{ui} + x_{vi} - p_{uvi} \leq 1, \qquad \forall u, v \in V, u < v, \ \forall i \in V,$$

$$x_{ui} \in \{0,1\}, \qquad \forall u \in V, \ \forall i \in V,$$

$$y_i \in \{0,1\}, \qquad \forall i \in V,$$

$$p_{uvi} \in \{0,1\}, \qquad \forall u, v \in V, u < v, \ \forall i \in V,$$

onde K_u representa o tráfego total na localidade u, ou seja, $K_u = \sum_{v \neq u} d_{uv}$.

A última direção futura é proposta para o projeto lógico do planejamento da rede *backbone*, mais especificamente, no que se refere ao problema de interconexão dos anéis. Propomos um estudo da estrutura facial do politopo associado ao problema do caixeiro viajante simétrico generalizado (PCVSG), visto que a resolução do problema de interconexão dos anéis corresponde à

resolução de uma instância do PCVSG. Um estudo inicial deste politopo pode ser encontrado em Fischetti *et al.* [21, 22].

Referências Bibliográficas

[1] Altinkemer, Kemal. "Topological Designing of Ring Networks". *Computers and Operations Research*, v. 21, n. 4, pp. 421–431, 1994.

[2] Appelgren, L. H. "A Column Generation Algorithm for a Ship Scheduling Problem". *Transportation Science*, v. 3, pp. 53–68, 1969.

[3] Aringhieri, R., Dell'Amico, M. *Solution of the SONET Ring Assignment Problem with Capacity Constraints.* Technical Report DISMI TR-12, Dipartimento di Scienze e Metodi dell'Ingegneria, Università di Modena e Reggio Emilia, 2001.

[4] Balas, E., Ceria, S., Cornuéjols, G., Natraj, N. "Gomory Cuts Revisited". *Operations Research Letters*, v. 19, pp. 1–9, 1996.

[5] Balas, E., Jeroslow, R. "Canonical Cuts on the Unit Hyperplane". *SIAM Journal on Applied Mathematics*, v. 23, n. 1, pp. 61–69, 1972.

[6] Balas, E., Padberg, M. "Set Partitioning: a Survey". *SIAM Review*, v. 18, pp. 710–760, 1976.

[7] Barahona, F., Mahjoub, A. R. "On the Cut Polytope". *Mathematical Programming*, v. 36, pp. 157–173, 1986.

[8] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Salvelsbergh, M. W. P., Vance, P. H. "Branch-and-Price: Column Generation for Solving Huge Integer Programs". *Operations Research*, v. 46, pp. 316–329, 1998.

[9] Bondy, J., Murty, U. S. R. *Graph Theory with Applications.* Macmillan, London, 1976.

[10] Boyd, E. Andrew. "Fenchel Cutting Planes for Integer Programs". *Operations Research*, v. 42, n. 1, pp. 53–64, 1994.

[11] Cormen, T., Leiserson, C., Rivest, R. *Introduction to Algorithms*. MIT Press, 1990.

[12] Cosares, S., Deutsch, D., Saniee, I., Wasem, O. "SONET Toolkit: a Decision Support System for Designing Robust and Cost-Effective Fiber-Optic Networks". *Interfaces*, v. 25, n. 1, pp. 30–40, 1995.

[13] Crowder, H., Johnson, E. L., Padberg, M. W. "Solving Large Scale Zero-One Linear Programming Problems". *Operations Research*, v. 31, pp. 803–834, 1983.

[14] Dash Optimization, Inc. *XPRESS-MP: User Manuals*, 1999. Versão 12.05.

[15] de Simone, C. "Lifting Facets on the Cut Polytope". *Operations Research Letters*, v. 9, pp. 341–344, 1990.

[16] de Souza, C. C. "Comunicação Privada", 2003.

[17] Desrosiers, J., Soumis, F., Desrochers, M. "Routing with Time Windows by Column Generation". *Networks*, v. 14, pp. 545–565, 1984.

[18] Farley, A. A. "A Note on Bouding a Class of Linear Programming Problems, Including Cutting Stock Problems". *Operations Research*, v. 38, pp. 922–923, 1990.

[19] Feo, T. A., Resende, M. G. C. "Greedy Randomized Adaptive Search Procedures". *Journal of Global Optimization*, v. 6, pp. 109–133, 1995.

[20] Ferreira, C. E., Wakabayashi, Y. *Combinatória Poliédrica e Planos-de-Corte Faciais*. X Escola de Computação, Campinas, SP, Brasil. Instituto de Computação, Unicamp, 1996.

[21] Fischetti, M., Salazar, J. J., Toth, P. "The Symmetric Generalized Traveling Salesman Problem". *Networks*, v. 26, pp. 113–123, 1995.

[22] Fischetti, M., Salazar, J. J., Toth, P. "A Branch-and-Cut Algorithm for the Symmetric Generalized Traveling Salesman Problem". *Operations Research*, v. 45, n. 3, pp. 378–394, 1997.

[23] Fortz, B., Soriano, P., Wynants, C. *A Tabu Search Algorithm for Self-Healing Ring Network Design.* Technical Report IS-MG 2000/15, Institut de Statisque, Université Libre de Bruxelles, 2000.

[24] Garey, M. R., Johnson, D. S. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[25] Gendreau, M., Labbé, M., Laporte, G. "Efficient Heuristics for the Design of Ring Networks". *Telecommunication Systems*, v. 4, pp. 177–188, 1995.

[26] Goldschmidt, O., Laugier, A., Olinick, E. V. *SONET/SDH Ring Assignment with Capacity Constraints.* Technical Report, Southern Methodist University, Department Computer Science and Engineering, September 2001.

[27] Gomory, R. E. "Solving Linear Programming Problems in Integer". In Bellman, R. E., M. Hall, Jr. (eds), *Combinatorial Analysis American*, pp. 211–216. American Mathematical Society, 1960.

[28] Hansen, P., Jaumard, B., de Aragão, M. Poggi. "Mixed-Integer Column Generation Algorithms and the Probabilistic Maximum Satisfiability Problem". In *II Conference on Integer Programming and Combinatorial Optimization - IPCO II*, pp. 165–180. Carnegie-Mellon University, 1992.

[29] Holm, S., Sorensen, M. M. "The Optimal Graph Partitioning Problem: Solution Method Based on Reducing Symmetric Nature and Combinatorial Cuts". *O.R. Spektrum*, v. 15, pp. 1–8, 1993.

[30] Jeong, G. W., Lee, K., Park, S., Park, K. "A Branch-and-Price Algorithm for the Steiner Tree Packing problem". *Computers and Operations Research*, v. 29, n. 3, pp. 221–241, 2002.

[31] Johnson, E. L. "Modelling and Strong Linear Programs for Mixed Integer Programming". In (editor), S. W. Wallace (editor), *Algorithms and Model Formulations in Mathematical Programming*, pp. 1–43. Springer-Berlin, 1989.

[32] Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P. "Progress in Linear Programming-Based Algorithms for Integer Programming: an Exposition". *INFORMS Journal on Computing*, v. 12, n. 1, pp. 2–23, 2000.

[33] Jünger, M., Reinelt, G., Rinaldi, G. "The Traveling Salesman Problem". In M. Ball, C.L. Monma, T. Magnanti, (eds.), G. Nemhauser (eds), *Networks Models*, Handbook on Operations Research and Management Science, pp. 225–330. Elsevier, North Holland, 1995.

[34] Kernighan, B. W., Lin, S. "An Efficient Heuristic Procedure for Partitioning Graphs". *Bell Systems Technical Journal*, v. 49, pp. 291–307, 1970.

[35] Klincewicz, J. G., Luss, H., Yan, D. C. K. "Designing Tributary Networks with Multiple Ring Families". *Computers and Operations Research*, v. 25, n. 12, pp. 1145–1157, 1998.

[36] Laguna, Manuel. "Clustering for the Design of SONET Rings in Interoffice Telecommunications". *Management Science*, v. 40, n. 1, pp. 1533–1541, 1994.

[37] Laporte, G., Nobert, Y. "Generalized Traveling Salesman Problem through n Sets of Nodes: an Integer Programming". *Information Systems and Operational Research*, v. 21, pp. 61–75, 1983.

[38] Lasdon, L. S. *Optimization Theory for Large Systems.* Macmillan, London, 1970.

[39] Lee, C. Y., Koh, S. J. "A Design of the Minimum Cost Survivability: a Tabu Search Approach". *Computers and Operations Research*, v. 24, n. 9, pp. 883–897, 1997.

[40] Lee, Y., Sherali, H., Han, J., Kim, S. "A Branch-and-Cut Algorithm for Solving an Intra-Ring Synchronous Optical Network Design Problem". *Networks*, v. 35, n. 3, pp. 223–232, 2000.

[41] Lim, S. K., Kim, Y.D. "An Integrated Approach to Dynamic Plant Location and Capacity Planning". *Journal of the Operational Research Society*, v. 50, pp. 1205–1216, 1999.

[42] Lim, S. K., Kim, Y.D. "Plant Location and Procurement Planning in Knockdown Production Systems". *Journal of the Operational Research Society*, v. 52, pp. 271–282, 2001.

[43] Lin, S., Kernighan, B. W. "An Efficient Heuristic Algorithm for the Travelling-Salesman Problem". *Operations Research*, v. 21, pp. 498–516, 1973.

[44] Luss, H., Rosenwein, M. B., Wong, R. T. "Topological Network Design for SONET Ring Architecture". *IEEE Transactions Systems, Man and Cybernetics - part A: Systems and Humans*, v. 28, n. 6, pp. 780–790, November 1998.

[45] Macambira, E. M., Maculan, N., de Souza, C. C. "Problema de Atribuição de Localidades a Anéis SONET: Formulação, Desigualdades Válidas e Facetas". XXV Congresso Nacional de Matemática Aplicada e Computacional, Nova Friburgo, RJ, Brasil, 2002.

[46] Macambira, E. M., Maculan, N., de Souza, C. C. "Problema de Atribuição de Localidades a anéis SONET: Formulações Inteiras 0-1 e Experimentos Computacionais". XI Congresso Latino-Iberoamericano de Investigação Operacional, Concépcion, Chile, 2002.

[47] Macambira, E. M., Maculan, N., de Souza, C. C. "Reduzindo a Simetria no Problema de Atribuição de Localidades a Anéis SONET com o Emprego de Planos-de-Corte". XXV Congresso Nacional de Matemática Aplicada e Computacional, Nova Friburgo, RJ, Brasil, 2002.

[48] Maculan, N., Passini, M. M., Brito, J. A. M., Lisser, A. "Column Generation Method for Network Design". In Gendreau, M., (eds.), P. Marcotte (eds), *Transportation and Network Analysis: Current Trends*, Chapter 11. Kluwer Academic Publishers, 2002.

[49] Maculan, N., Plateau, G., Loiseau, I. "Column Generation Technique in Network Design". In *IV ALIO/EURO Workshop on Applied Combinatorial Optimization*, pp. 77–82. Universidad de Concepción and Catholic University of Rio de Janeiro, 2002. November 4-6.

[50] Maculan, Nelson. "Linear 0-1 Programming". In e M. Resende, P. Pardalos (editor), *Handbook of Applied Combinatorial Optimization.* Oxford University Press, 2002.

[51] Magnanti, T. L., Wong, R. T. "Network Design and Transportation Planning: Models and Algorithms". *Transportation Science*, v. 18, pp. 1–55, 1984.

[52] Martello, S., Toth, P. *An Algorithm for the Generalized Assignment Problem*, pp. 589–603. Operational Research, volume 81. North Holland, Amsterdam, 1981.

[53] Mehrotra, A. "Cardinality Constrained Boolean Quadratic Polytope". *Discrete Applied Mathematics*, v. 79, pp. 137–154, 1997.

[54] Mehrotra, A., Trick, M. A. "Cliques and Clustering: a Combinatorial Approach". *Operations Research Letters*, v. 22, pp. 1–12, 1998.

[55] Maculan, N., Macambira, E. M., de Souza, C. C. *Planos-de-corte Geométricos Aplicados a Problemas de Programação Linear 0-1.* Relatório Técnico ES-574/02, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, 2002.

[56] Maculan, N., Macambira, E. M., Simonetti, L. G. "Geometrical Efficient Cuts in Integer 0-1 Programming". XVIII European Conference on Operational Research, Rotterdam, Holanda, 2001.

[57] Maculan, N., Macambira, E. M., de Souza, C. C. *Geometrical Cuts for 0-1 Integer Programming.* Technical Report IC-02-006, Instituto de Computação, Universidade Estadual de Campinas, 2002. Submetido à revista RAIRO - Operations Research em Abril de 2002.

[58] Maculan, N., Macambira, E. M, Simonetti, L. G., de Souza, C. C. *Resolução do Problema do Caixeiro Viajante Simétrico com o Emprego de Planos-de-corte Geométricos.* Relatório Técnico ES-573/02, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, 2002.

[59] Nemhauser, G. L., Wolsey, L. *Integer and Combinatorial Optimization.* Wiley & Sons, 1988.

[60] Padberg, M. "The Boolean Quadric Polytope: Some Characteristics, Facets and Relatives". *Mathematical Programming*, v. B-45, pp. 139–172, 1989.

[61] Reinelt, Gerhard. "TSPLIB: a Traveling Salesman Problem Library". *ORSA Journal on Computing*, v. 3, pp. 376–384, 1991.

[62] Roy, T. J. Van, Wolsey, L. A. "Solving Mixed 0-1 Problems by Automatic Reformulation". *Operations Research*, v. 35, pp. 45–57, 1987.

[63] Ryan, D. M., Foster, B. A. "An Integer Programming Approach to Scheduling". In (editor), A. Wren (editor), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280. Noth-Holland, 1981.

[64] Savelsbergh, M.W.P. "A Branch-and-Price Algorithm for the Generalized Assignment Problem". *Operations Research*, v. 45, pp. 831–841, 1997.

[65] Sherali, H. D., Lee, Youngho, Adams, W. P. "A Simultaneous Lifting Strategy for Identifying New Classes of Facets for the Boolean Quadric Polytope". *Operations Research Letters*, v. 17, pp. 19–26, 1995.

[66] Sherali, H. D., Smith, J. C. "Improving Discrete Model Representations via Symmetry Considerations". *Management Science*, v. 47, n. 10, pp. 1396–1407, 2001.

[67] Sol, M., Savelsbergh, M.W.P. "DRIVE: Dynamic Routing of Independent VEhicles". *Operations Research*, v. 46, pp. 474–490, 1998.

[68] Soriano, P., Wynants, C., Séguin, R., Labbé, M., Gendreau, M., Fortz, B. "Design and Dimensioning of Survivable SDH/SONET Networks". In e P. Soriano (eds.), B. Sansò (editor), *Telecommunications Network Planning*, pp. 147–167. Kluwer Academics Publishers, 1999.

[69] Sosnosky, J., ho Wu, Tsong. "SONET Ring Applications for Survivable Fiber Loop Networks". *IEEE Communications Magazine*, v. 29, n. 6, pp. 51–58, June 1991.

[70] Sutter, A., Vanderbeck, F., Wolsey, L. "Optimal Placement of ADD/DROP Multiplexers: Heuristic and Exact Algorithms". *Operations Research*, v. 46, n. 5, pp. 719–728, 1998.

[71] Thienel, Stefan. *ABACUS: A Branch-And-CUt Systems.* PhD thesis, Mathemastisch Naturwissenschaftlichen Fakultät, Universität zu Köln, 1995.

[72] Vanderbeck, F. "Computational Study of a Column Generation Algorithm for Bin Packing and Cutting Stock Problems". *Mathematical Programming*, v. 86, n. 3, pp. 565–594, 1999.

[73] Vanderbeck, F. "On Dantzig-Wolfe Decomposition in Integer Programming and Ways to Perform Branching in a Branch-and-Price Algorithm". *Operations Research*, v. 48, n. 6, pp. 915–926, 2000.

[74] Vanderbeck, F., Wolsey, L. A. "An Exact Algorithm for IP Column Generation". *Operations Research Letters*, v. 19, n. 4, pp. 151–159, 1996.

[75] Vanderbeck, François. *Decomposition and Column Generation for Integer Programs.* PhD thesis, Faculté des Sciences Appliquées, Université Catholique de Louvain, Septembre 1994.

[76] Wasem, O. J., Wu, Tsong-Ho, Cardwell, R. H. "Survivable SONET Networks - Design Methodology". *IEEE Journal on Selected Areas in Communications*, v. 12, n. 1, pp. 205–212, January 1994.

[77] West, D. B. *Introduction to Graph Theory.* Prentice-Hall, 2000.

[78] Wolsey, L. A. "Faces for Linear Inequality in 0-1 Variables". *Mathematical Programming*, v. 8, pp. 165–178, 1975.

[79] Wolsey, L. A. "Facets and Strong Valids Inequalities for Integer Programs". *Operations Research*, v. 24, pp. 367–372, 1976.

[80] Wolsey, L. A. *Integer Programming.* Wiley & Sons, 1998.

Índice Remissivo

–A–

algoritmo
  branch-and-cut, 95
  branch-and-price, 78, 95, 111
  cut-and-branch, 149, 154, 157
anel
  capacidade, 12
  federal, 3, 11

–C–

complexidade
  NP-difícil, 14
componentes
  conexas de um grafo, 155
conjunto inicial de colunas, 96

–D–

desigualdade
  árvore, 92
  2-*matching*, 155
  clique, 92
  clique triangular, 45
  clique triangular generalizada, 48
  de eliminação de subciclos, 145, 146, 152, 155
  triangular, 91
detecção de *tailing off*, 102
dimensão
  cheia, 34, 90
distância de Hamming, 131, 135, 137, 139, 141, 142

–E–

estratégias
  de ramificação, 78, 99
  de seleção de colunas, 102
  de vizinhança, 64

–F–

fase
  de busca local, 66, 69
  de construção, 65, 68
formulação
  de empacotamento, 14, 19, 32, 82
  de particionamento, 14

–G–

geração de colunas, 78

–H–

heurística, 63
cut-based, 63
edge-based, 63
node-based, 64
primal, 101

–I–

instâncias
aleatórias, 25, 28, 71, 113, 120
geométricas, 23, 28, 71, 112, 119

–L–

lista restrita de candidatos, 66, 68

–M–

metaheurística, 64
GRASP, 65, 71
busca tabu, 64
movimentos de melhoria
de Lin-Kernighan, 69, 152

–N–

nó
ativo, 98
inativo, 98

–P–

planejamento
de redes de telecomunicações, 2, 6
de uma rede *backbone*, 2, 3, 6, 11
plano-de-corte
p-cilíndrico, 141, 143
p-esférico, 137, 143, 152, 153, 155, 157, 160
1-cilíndrico, 140
1-esférico, 134, 135
cilíndrico, 138
esférico, 134
planos-de-corte
canônicos, 131, 143, 144
geométricos, 130, 142
problema
problema auxiliar, 79, 85, 87, 88, 94, 95
booleano quadrático, 93
de atribuição de localidades a anéis, 6, 11–13, 16, 32
de atribuição generalizado, 19
de construção dos anéis idênticos, 6, 148, 160
de particionamento de um grafo, 4
do caixeiro viajante simétrico, 6, 160
do caixeiro viajante simétrico generalizado, 6
do corte ótimo de um grafo, 93
mestre, 79, 85, 87
procedimento *branch-and-bound*, 53, 56, 78, 98, 156

procedimentos
de perturbação, 69
de *bounding*, 100
processo
de linearização, 16
projeto
físico, 3, 4, 6
lógico, 4, 6

–R–

rede
backbone, 1
de acesso local, 1
de distribuiçã, 1
de telecomunicações, 1
reformulação, 80, 81
regra
de Ryan-Foster, 99
relaxação 2-*matching*, 155, 157
rotina
de separação, 155

–S–

seleção de nós, 98
simetria
das soluções, 32, 55, 56
sobrevivência
de uma rede, 2

–T–

tecnologia
SONET, 2, 11
topologia
em anel, 3, 12, 13
tráfego
externo, 67
interno, 67
total, 67

Notação e Símbolos

ADM : *Add-Drop Multiplexers*;

DCS : *Digital Cross-connect System*;

STM : *Synchronous Transport Module*;

SONET : *Synchronous Optical Network*;

PL : Problema Linear;

PPLI 0-1 : Problema de Programação Linear Inteiro no qual todas as variáveis assumem valores iguais a 0 ou 1;

RL : Relaxação Linear;

Mbs : Mega bits por segundo;

B : capacidade de um anel em Mbs;

L : conjunto de localidades;

$G = (V, E)$: grafo com um conjunto V de vértices e um conjunto E de arestas;

K_n : grafo completo com n vértices;

(u, v) : uma aresta pertencente ao conjunto E;

d_{uv} : demanda da aresta (u, v);

V_i : subconjunto de elementos do conjunto V;

$E(S)$: conjunto de arestas com ambos os extremos em S;

$E(V \setminus S)$: conjunto de arestas com um extremo em V e o outro extremo em S;

$|S|$: número de elementos no conjunto S;

$\mathcal{N}$: nó raiz da árvore *branch-and-bound*;

SRAP : Problema de Atribuição de Localidades a Anéis;

PCVS : Problema do Caixeiro Viajante Simétrico;

PCVSG : Problema do Caixeiro Viajante Simétrico Generalizado;

C1 : classe de instâncias geradas por Goldschmidt *et al.* para o SRAP;

C2 : classe de instâncias geradas por Aringhieri e Dell'Amico para o SRAP;

(P1) : formulação de particionamento;

(P2) : formulação de empacotamento;

$\mathbb{R}^{|S|}$: conjunto de vetores reais definido num espaço de dimensão $|S|$;

$\mathbb{B}^{|S|}$: conjunto de vetores 0-1 definido num espaço de dimensão $|S|$;

0^k : vetor nulo em um espaço de dimensão k;

e_i : vetor em que todas as componentes são nulas, excetuando-se a i-ésima componente, que vale 1;

e_i^k : vetor num espaço de dimensão k com todas as componentes nulas, excetuando-se a i-ésima componente, que possui valor 1;

$e_{ij}^{k \times t}$: vetor num espaço de dimensão $k \times t$ com todas as componentes são nulas, excetuando-se a $(k \times i + j)$-ésima componente, que vale 1;

$\mathcal{X}^*$: solução fracionária ótima;

$x_i(S)$: $\sum_{u \in V(S)} x_{ui}$;

$p_i(S)$: $\sum_{u,v \in V(S), u<v} p_{uvi}$;

$\mathcal{Q}$: politopo descrevendo a envoltória convexa dos vetores de incidência de partições viáveis para o SRAP, isto é, a partição de G em agrupamentos, tal que a capacidade destes agrupamentos seja no máximo igual a B;

LRC : Lista Restrita de Candidatos;

I_r : tráfego interno no anel r;

E_r : tráfego externo no anel r;

T_r : tráfego total no anel r;

F : tráfego total no anel federal;

$BFed$: quantidade de tráfego em excesso no anel federal;

$BNet$: quantidade de tráfego em excesso nos anéis de uma solução;

D : demanda total na rede;

D_j : soma das demandas dentro do agrupamento j;

W_j : soma das demandas nas arestas do corte V_j;

(P3) : reformulação do SRAP;

μ_i : variáveis duais associadas às restrições (5.14);

π : variáveis duais associadas às restrições (5.15);

$\bar{c}_j$: custo reduzido do agrupamento V_j;

(P4) : problema mestre;

(P5) : problema auxiliar;

$\mathcal{P}'$: politopo descrevendo a envoltória convexa das soluções inteiras do sistema (5.18)–(5.23);

`CUT` : Problema do Corte Ótimo de um grafo;

`BQ` : Problema Booleano Quadrático;

$\mathcal{P}^{\mathcal{C}}$: politopo associado ao problema CUT;

$\mathcal{P}^{\mathcal{B}}$: politopo associado ao problema PBQ;

(P6) : problema de programação linear;

(P7) : Problema de Programação Linear Inteira 0-1 para o Problema do Caixeiro Viajante Simétrico;

$dist(x, y)$: distância de Hamming entre os pontos x e y;

$W_p(\overline{x})$: conjunto de todos os pontos em $\mathbb{B}^n$ que possuem distância de Hamming no máximo igual a $p-1$, com relação ao ponto $\overline{x}$;

$W_p(\overline{x}^1, \overline{x}^2)$: conjunto de pontos em $\mathbb{B}^n$, cuja distância de Hamming, com relação a $\overline{x}^1$ ou a $\overline{x}^2$, é estritamente menor do que p;

$\mathcal{S}_1(\overline{x})$: combinação convexa de todos os elementos do conjunto $X_1(\overline{x}) = \mathbb{B}^n \setminus \{\overline{x}\}$, ou seja, a envoltória convexa de $X_1(\overline{x})$;

$\mathcal{S}_p(\overline{x})$: combinação convexa de todos os elementos do conjunto $X_p(\overline{x}) = \mathbb{B}^n \setminus W_p(\overline{x})$, ou seja, a envoltória convexa de $X_p(\overline{x})$;

$\mathcal{C}_p(\overline{x}^1, \overline{x}^2)$: combinação convexa de todos os elementos do conjunto $X_p(\overline{x}^1, \overline{x}^2) = \mathbb{B}^n \setminus W_p(\overline{x}^1, \overline{x}^2)$, ou seja, a envoltória convexa de $X_p(\overline{x}^1, \overline{x}^2)$;

$\mathcal{H}$: conjunto de vértices do hipercubo unitário, num espaço com dimensão $|S|$;

X^k : conjunto de pontos em $\mathcal{H}$ que possuem exatamente i componentes iguais;

$\mathcal{K}$: conjunto de índices em $\{1, \ldots, n\}$, para os quais as componentes dos pontos em X^k são iguais;

$\mathcal{H}_n$: hipercubo unitário;

$\mathcal{F}^k$: face do hipercubo unitário $\mathcal{H}_n$, com dimensão igual a k;

$\mathcal{H}(\mathcal{F}^k)_d$: hiperplano canônico de ordem k, associado à face $\mathcal{F}^k$ do hipercubo unitário $\mathcal{H}_n$, com $d \in \{0, 1, \ldots, n-k\}$.